Колисниченко Д.Н.

Linux на примерах

ПРАКТИКА, ПРАКТИКА И ТОЛЬКО ПРАКТИКА

- Основы Linux
- Практические примеры
- Администрирование и команды Linux
- Хаки Linuxı Lı

Колисниченко Д. Н.

LINUX НА ПРИМЕРАХ

Практика, практика и только практика

"Издательство Наука и Техника"

Санкт-Петербург

УДК 004.42
ББК 32.973

Колисниченко Д. Н.

LINUX на примерах. Практика, практика и только практика — СПб.: Издательство Наука и Техника, 2022. — 320 с., ил.

ISBN 978-5-94387-410-9

Данная книга является практическим руководством по работе в Linux и ее администрированию. Книга содержит в себе как теоретические, так и практические материалы, т.е. теория и практика объединены в одно целое. Изложение ведется с учетом самых разных дистрибутивов Linux.

Будет рассмотрен широкий спектр задач и возможностей Linux – от самых основ (установка системы, вход и завершение работы, настройка системы, основы командной строки) до более продвинутых тем (локальное администрирование в Linux; управление файловой системой; маршрутизация и настройка брандмауэра; системные процессы и т.д.)

Книга будет полезна как для тех, кто только заинтересовался Линуксом, так и для тех, кто хочет расширить свои навыки использования этой операционной системой.

ISBN 978-5-94387-410-9

9 78- 5- 94387- 410- 9

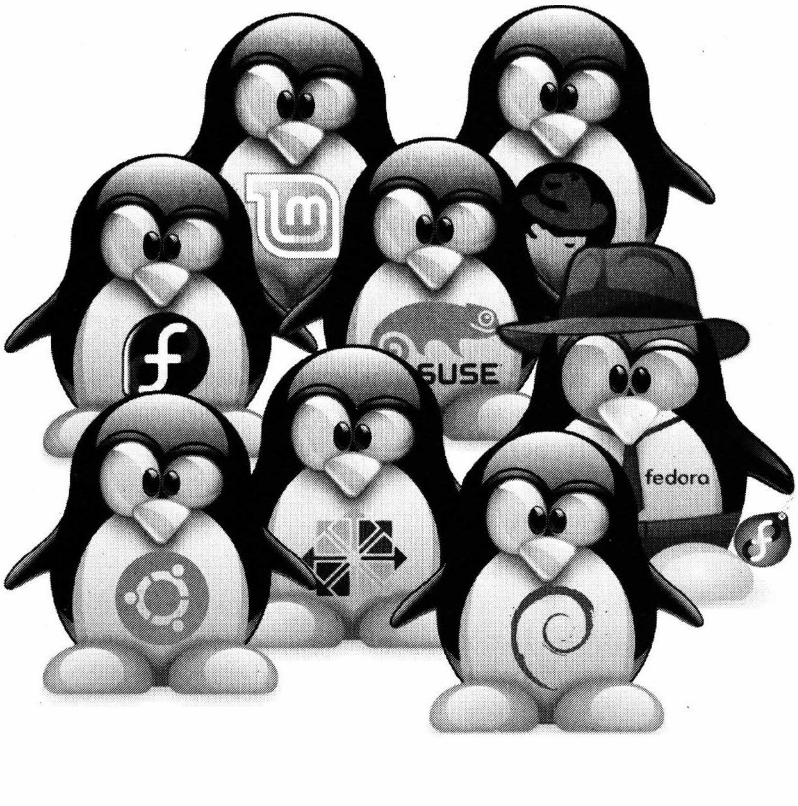
Контактные телефоны издательства:

(812) 412 70 26

Официальный сайт: www.nit.com.ru

Содержание

Введение



Linux – удивительная операционная система. Впервые она появилась в 1991 году, а пик ее популярности приходится на начало 2000-х. Затем интерес к ней пошел на спад и можно было подумать, что через пару лет о ней никто не вспомнит. Но Linux подобен Фениксу, восставшему из пепла. Интерес к ней появился не только у обычных пользователей (иначе бы я сейчас не писал эту книгу), но и крупных корпораций, таких как Microsoft, IBM и т.д.

Некоторые из книг бывают сугубо практическими, некоторые – сугубо теоретическими. Книга, которую вы держите в руках, будет эдаким симбиозом и содержать в себе, как теоретические, так и практические материалы. Теория и практика будут объединены в одно целое – не будет отдельных больших и скучных теоретических глав.

Книга ориентирована на пользователей разного уровня, но уже подготовленным пользователям (например, квалифицированным Windows-пользователям или администраторам), которым по долгу службы или ради личного интереса предстоит разбираться с Linux, читать книгу будет немного проще.

Графический интерфейс практически не рассматривается в этой книге – по той лишь причине, что по простоте использования он уже дорос до Windows

– все наглядно, просто и понятно. Зато мы рассмотрим практические нюансы настройки операционной системы – от ее установки на локальный компьютер до добавления второго жесткого диска – в общем все, что нужно обычному пользователю для его обычного (домашнего или офисного) использования в этой книге есть.

Далее мы рассмотрим выбор дистрибутива Linux. Ведь Linux – это только "собирательное название", а по факту выбирать приходится между дистрибутивами Linux, которых очень и очень много. С 2001 года, согласно ресурсу Distrowatch, было создано почти 800 дистрибутивов… Понятно, что количество разрабатываемых дистрибутивов постоянно сокращается – слабые уходят с рынка, но выбор по-прежнему огромен. Несколько сотен дистрибутивов. Только на главной странице Distrowatch есть список TOP-100 – он обновляется ежедневно. Следовательно, в мире есть как минимум 100 активных дистрибутивов, которыми пользуются люди…

Самая первая версия Linux, появившаяся в 1991 году, представляла собой ядро и несколько приложений. В ней запускались компилятор **gcc** и командный интерпретатор **bash**. Поставлялась эта версия Linux в виде двух дискет – на первой было ядро, на второй – корневая файловая система с приложениями. Загружалась она тоже специфически – сначала нужно было вставить в дисковод первую дискету и дождаться, пока загрузится ядро, затем был запрос на вставку второй дискеты – с корневой файловой системой.

Первые дистрибутивы появились в 1992 году. Тогда отдельные энтузиасты или группы энтузиастов выпускали разные дистрибутивы (каждый, естественно, под своим именем). Грубо говоря, они отличались второй дискетой, на которой был немного другой набор программ. Далее, с развитием Linux, необходимые программы уже не помещались на одну дискету и пришлось устанавливать Linux на жесткий диск. Появились первые программы установки.

Чем отличаются разные дистрибутивы, кроме, разумеется, названия? Во-первых, у каждого дистрибутива своя программа установки (если не считать дистрибутивов-клонов, которые заимствуют инсталлятор у родительского дистрибутива, меняя только название дистрибутива). Во-вторых, у каждого будет свой набор программ – на усмотрение разработчика. По сути, с 1992 года ничего не поменялось.

Если копнуть дальше, вникнуть в набор программ, то начинаешь видеть более глубинные изменения, например, разницу в менеджере пакетов и системе инициализации. По сути, что менеджер пакетов, что система инициализации – это тоже программы. Но программа программе – рознь.

Сейчас мы не будем вникать во всех технические тонкости. Для этого есть соответствующие главы это книги. Лучше рассмотрим актуальные на данный момент дистрибутивы.

На данный момент Linux-пользователям доступно несколько основных дистрибутивов:

- **Debian** – тот самый надежный Debian, появившийся в 1993 году. Это единственный широко распространенный дистрибутив, доживший до наших дней под оригинальным названием.
- **Fedora** – потомок популярного ранее дистрибутива Red Hat, существование которого было прекращено в 2004 году. Тогда пользователям предоставили выбор: либо они мигрируют на корпоративный (коммерческий) RHEL (Red Hat Enterprise Linux), либо на бесплатный Fedora (ранее Fedora Core). На данный момент Fedora – развивающийся дистрибутив, последняя рабочая версия которого вышла 2 ноября 2021 года, а выпуск новых версий производится каждые 6-8 месяцев.
- **Ubuntu** – изначально основан на Debian, первая версия появилась в 2004 году, последняя актуальная – 14 октября 2021 года (версия 21.10). Обновляется каждые 6 месяцев. Как и Fedora, имеет несколько вариантов, в том числе серверный. Популярным неофициальным (не от разработчиков Ubuntu) форком является дистрибутив Mint – "доведенная до ума" версия Ubuntu.
- **openSUSE** – изначально основан на дистрибутиве Slackware и первая его версия вышла в октябре 2005 года (сравнительно молодой дистрибутив). На данный момент доступна рабочая версия от 2 июня 2021 года, а обновляется дистрибутив примерно раз в год. В отличие от Ubuntu, использует систему пакетов RPM, что делает его ближе к Fedora – со временем в состав openSUSE включили некоторые решения из Red Hat – систему пакетов RPM, использование sysconfig – что сделало больше похожим на Red Hat, чем на Slackware.
- **ALT Linux** – как ни крути, но этот отечественный дистрибутив заслуживает уважения – хотя бы за то, что дожил до наших дней и не развалился, как многие другие. И учтите, первая его версия появилась в 1999 году (то есть ему больше 20 лет), а не в 2004-2005, как Ubuntu и openSUSE. Последняя версия от 16 декабря 2021 года.
- **CentOS** (*Community ENTerprise Operating System*) – общественная корпоративная операционная система. Основан на RHEL и совместим с ним.

Содержит из свободного ПО с открытым кодом. Первая версия вышла в 2004 году, на данный момент последней актуальной версией является версия от 16 ноября 2021 года – дистрибутив развивается. Дистрибутив очень надежный – иного от корпоративной ОС и не следует ожидать, пусть и не содержит самых новых пакетов ПО, как, например, Fedora.

- **Astra Linux** – дистрибутив специального назначения на базе ядра Linux, созданная для комплексной защиты информации и построения защищенных автоматизированных систем. Сертифицирована в системах сертификации средств защиты информации Минобороны, ФСТЭК и ФСБ России. Первая версия увидела свет в 2009 году, а последняя рабочая версия вышла 7 сентября 2021 года.

Все эти дистрибутивы на момент написания этих строк (начало 2022 года) обновлялись в прошлом году (2021 год), что говорит о том, что разработчики дистрибутив не забросили, и он продолжает развиваться. Бывает так, что по тем или иным причинам поддержка того или иного дистрибутива прекращается. Так случилось с BlackCat Linux, Mandriva и многими другими дистрибутивами.

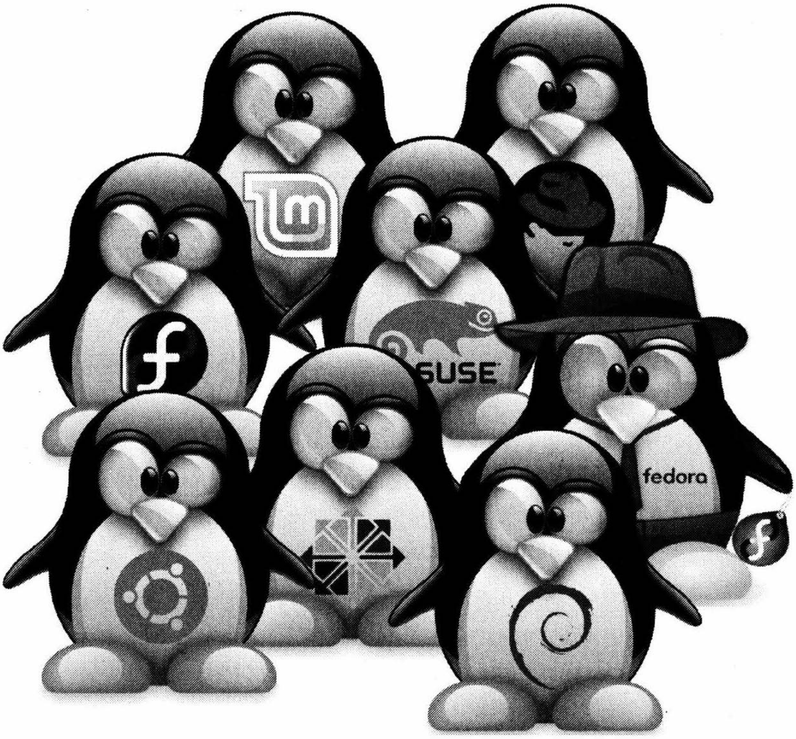
Какой дистрибутив выбрать? Вопрос довольно распространенный, но однозначного ответа на него нет. Все зависит от применения и личных предпочтений. Например, фанатов Ubuntu ни за что не заставишь установить Fedora и наоборот. Если же у вас своего мнения относительно дистрибутива не сформировалось, то можно выбирать один из следующих дистрибутивов – Fedora, CentOS, Ubuntu, Debian. На сервере я бы рекомендовал более стабильные CentOS и Debian, но поскольку вы только начинаете разбираться с Linux, можно смело использовать Fedora и Ubuntu. С ними вам будет проще и они более универсальные. Оба дистрибутива смело подойдут как для рабочей станции (или домашнего компьютера), так и для сервера.

Когда же речь заходит о работе с персональными данными, то ненароком вмешивается закон ФЗ-152 и расставляет все на свои места. Обработка и хранение таких данных должно происходить только с использованием сертифицированного ПО. Для Windows достаточный большой выбор различного сертифицированного ПО – и программы для шифрования, и антивирусы, и брандмауэры и т.д. Выходит, если вы создаете Интернет-магазин, что подразумевает хранение и обработку персональных данных, то вы или ограничены Windows (что не хочется), или же нужно смотреть в сторону сертифицированных дистрибутивов Linux. Такими являются Astra Linux и ALT Linux. Здесь уже выбирается не по личным предпочтениям, а из-

за "бумажки". Мы не будем рассматривать использование Linux на предприятии, а поговорим об использовании этого дистрибутива на домашнем компьютере – исключительно для домашнего применения. Свой выбор остановим на Ubuntu 21.10 – самой актуальной версии Ubuntu на данный момент. А на предприятии, как правило, есть высокооплачиваемый специалист-администратор, который сможет без проблем разобраться с любым дистрибутивом Linux.

Глава 1.

Установка системы



Несмотря на то, что у всех дистрибутивов Linux собственный инсталлятор, свой интерфейс пользователя, разный набор программного обеспечения, устанавливаемого по умолчанию, все они устанавливаются по единому принципу. В этой главе мы рассмотрим попарно установку Ubuntu и Astra Linux.

1.1. Загрузка с инсталляционного диска

Первое с чего нужно начинать установку системы – с получения инсталляционного носителя. Поскольку рассматриваемые дистрибутивы Linux распространяются абсолютно бесплатно, нет никакого смысла загружать ISO-образы со сторонних ресурсов. В нашем случае необходимые ISO-файлы можно получить с сайтов https://ubuntu.com и https://astralinux.ru/.

Далее нужно создать сам инсталляционный носитель. В качестве такового носителя может выступать либо DVD-диск, либо USB-флешка. Эра DVD-дисков давно прошла (сейчас даже в продаже их найти сложно), а вот USB-флешка, как правило, всегда под рукой.

Для подготовки загрузочного USB-диска нам нужна флешка с размером от 4 Гб и компьютер под управлением Windows. Скачайте приложение Rufus (https://rufus.ie/) и выполните следующие действия:

1. Если на флешке есть данные, скопируйте их на жесткий диск, поскольку в процессе создания загрузочного носителя они будут уничтожены.

2. Запустите Rufus.
3. Из списка **Устройство** выберите флешку. Убедитесь, что вы выбрали правильную флешку, если подключено несколько устройств.
4. Нажмите кнопку **Выбрать** для выбора ISO-образа, который будет записан на флешку.
5. Остальные параметры оставьте как есть. Схема раздела должна быть MBR, целевая система – BIOS или UEFI, файловая система – FAT32, размер кластера – 4096.
6. Нажмите кнопку СТАРТ.
7. Дождитесь, пока программа запишет ISO-образ на флешку.
8. Подключите USB-флешку к целевому компьютеру (на который будет происходить установка Linux)
9. Перезагрузите целевой компьютер.
10. Войдите в BIOS SETUP (обычно для этого используется клавиша DEL или F2, но нужная комбинация может отличаться – обратитесь к руководству по материнской плате).
11. В качестве загрузочного устройства выберите созданную флешку.
12. Выйдите из BIOS SETUP с сохранением изменений.

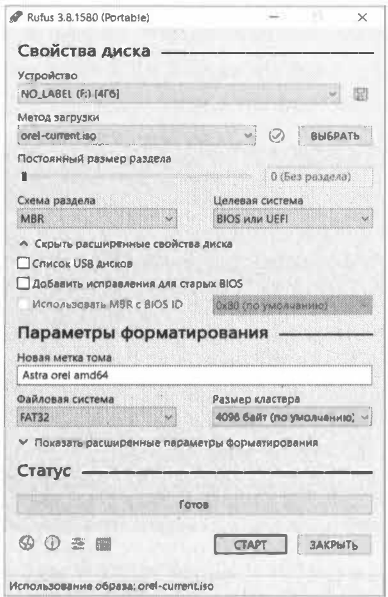


Рис. 1.1. Запись ISO образа с Linux на флешку

Если вы все сделали правильно, то увидите начальный экран загрузчика Linux.

1.2. Принцип установки Linux

Самое главное при установке Linux – выполнить разметку жесткого диска и не забыть установленный пароль, указанный при установке. Неправильная разметка жесткого диска означает, что в процессе работы с системой ее придется изменить, а это сделать не всегда просто, особенно, если сервер уже работает. Ну и пароль пользователя желательно тоже не забывать, иначе придется попотеть, чтобы взломать собственную же систему. А возможность такого взлома зависит, прежде всего, от настроек дистрибутива по умолчанию – в одних дистрибутивах все будет хорошо, а в других у вас ничего не получится. Хотя вряд ли можно назвать хорошим возможность взлома локального сервера (к которому у вас есть физический доступ) – так что все относительно.

1.3. Загрузка с инсталляционного носителя

В случае с Ubuntu 21.10 вы увидите вы увидите непрезентабельное меню загрузчика GRUB (рис. 1.2). Просто нажмите **Enter**. В предыдущих версиях меню было выполнено в фиолетовом цвете и позволяло выбрать язык программы установки. В самой последней версии язык выбирается после загрузки графического интерфейса (рис. 1.3).

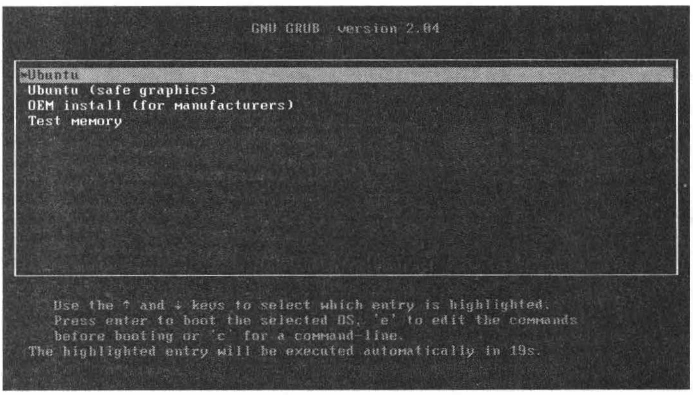


Рис. 1.2. Загрузчик GRUB (Ubuntu 21.10) – просто нажмите Enter

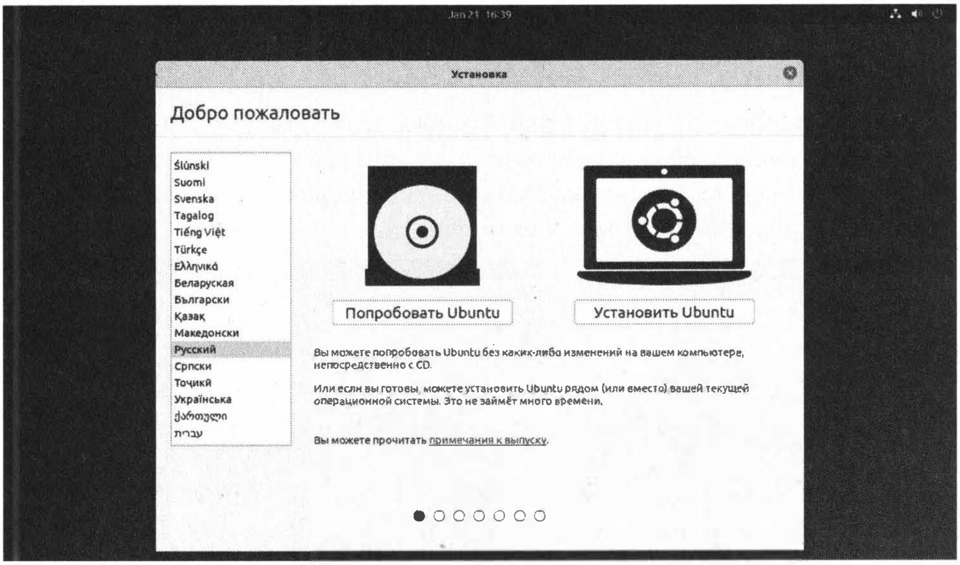


Рис. 1.3. Выбор языка после загрузки

В случае с Asta Linux танцев с бубном меньше – вы сразу видите начальное меню загрузчика и можете выбрать или графическую установку или установку в текстовом режиме (команду **Установка**), что подойдет для слабых компьютеров или для серверов, где графический интерфейс не нужен.

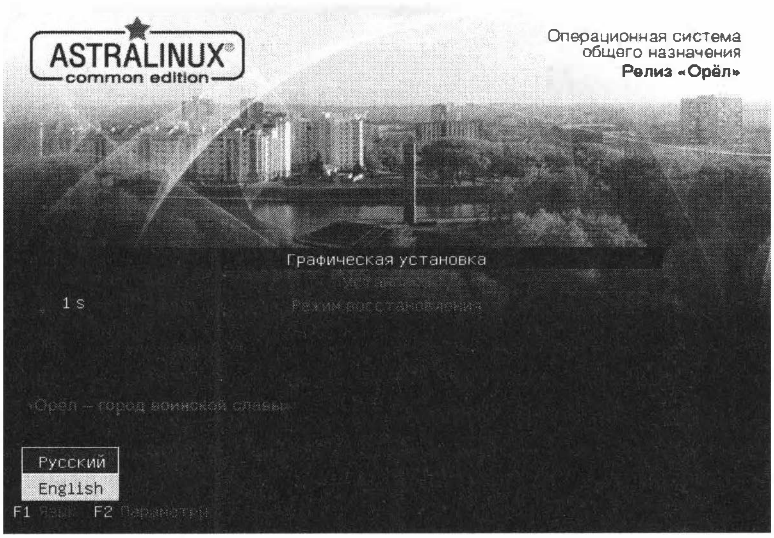


Рис. 1.4. Меню загрузчика Astra Linux

После выбора языка в Ubuntu вы можете или запустить ее в режиме LiveCD (кнопка **Попробовать Ubuntu**) или же в режиме установки (кнопка **Установить Ubuntu**). В режиме LiveCD можно работать с дистрибутивом как будто бы он уже установлен. Такой режим удобно использовать для устранения всяких неисправностей, которые возникают при работе с Linux или же если вы действительно хотите попробовать новую версию, чтобы понять, нужно ли вам ее устанавливать вместо уже имеющейся.

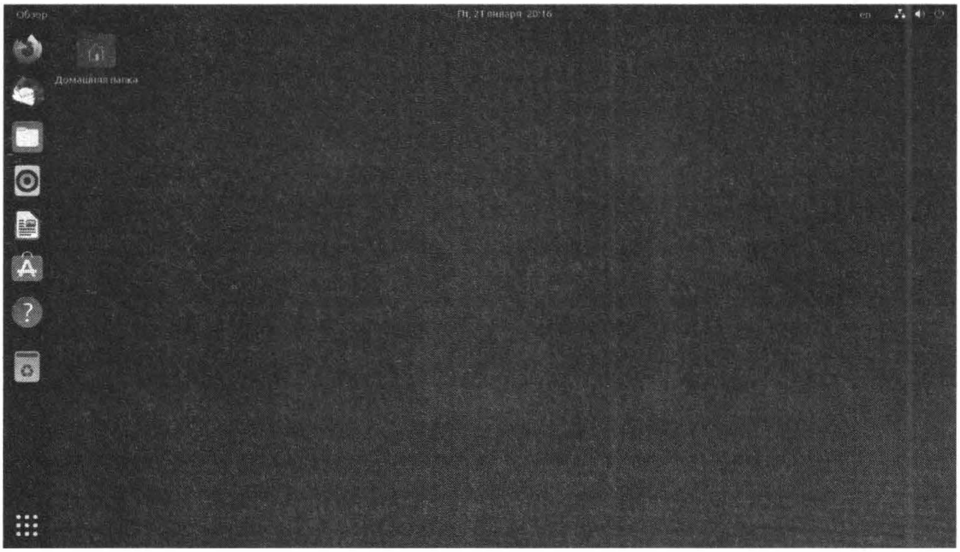


Рис. 1.5. Режим LiveCD Ubuntu 21.10

1.4. Начало установки

Сразу после запуска инсталлятора:

- Ubuntu: Нужно выбрать язык и для продолжения установки нажать кнопку **Установить Ubuntu**.
- Astra Linux: прочесть лицензионное соглашение и нажать кнопку **Продолжить**.

Дистрибутив Astra Linux основан на дистрибутиве Debian и использует такой же инсталлятор. В некоторых моментах он удобнее, чем инсталлятор Ubuntu, в некоторых – нет. Например, после нажатия кнопки начала установ-

ки в обоих случаях инсталляторы предложат выбрать раскладку клавиатуры (рис. 1.6, 1.7). Вот только в случае с Astra Linux вы можете выбрать комбинацию клавиш для переключения раскладки, что очень удобно. В Ubuntu вам придется это сделать после установки, если стандартная вам не подходит.

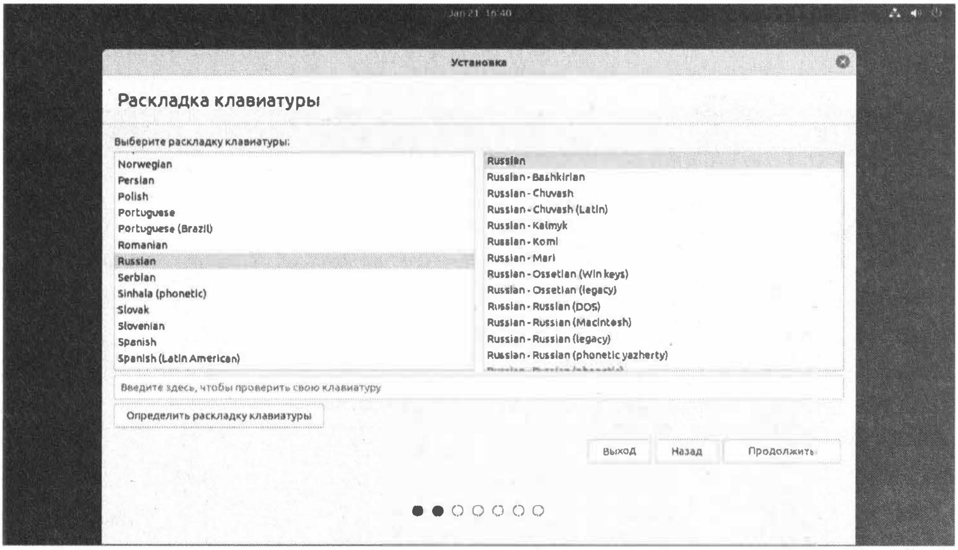


Рис. 1.6. Параметры клавиатуры (Ubuntu)



Рис. 1.7. Параметры клавиатуры (Astra Linux)

Сразу после выбора раскладки клавиатуры Ubuntu позволяет выбрать параметры программ. Как правило, оптимальные параметры показаны на рис. 1.8. Здесь используется обычная (а не минимальная) установка, все обновления загружаются во время установки, чтобы Ubuntu не надоедала вам с просьбами обновиться хотя бы после установки, а также мы устанавливаем сразу стороннее ПО для поддержки различных медиа-форматов (проще говоря, кодеки). В Astra Linux подобного окна нет.

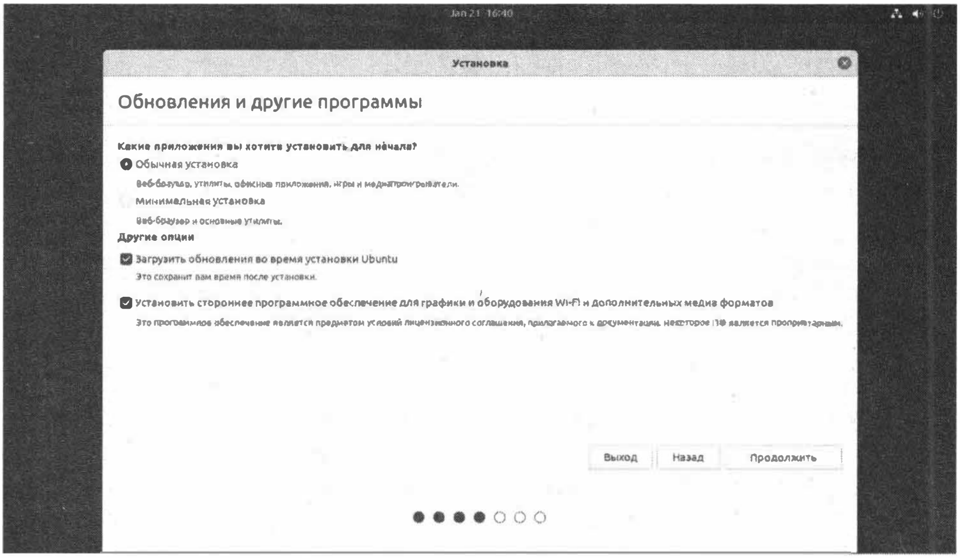


Рис. 1.8. Параметры программ

1.5. Разметка диска

1.5.1. Общие сведения о разметке диска

Один из самых важных моментов при установке любого дистрибутива Linux – это разметка жесткого диска. Если вы в процессе установки, например, забудете выбрать какой-то пакет – его очень просто установить после установки. Но вот если разметка диска будет выполнена неправильно, то исправить ситуацию будет гораздо сложнее. В некоторых случаях придется даже переустанавливать всю систему.

Linux использует свою файловую систему (некоторые дистрибутивы используют **ext4**, некоторые **xfs**, о файловой системе мы поговорим подробно в дальнейшем), поэтому ее нельзя установить в уже имеющиеся на жестком диске разделы. Если вы устанавливаете Linux не на новый жесткий диск,

тогда желательно удалить все разделы сторонних операционных систем. Не думаю, что на сервере будут сожительствовать две операционных системы. Понятно, что это не нужно делать на домашнем компьютере, где, скорее всего, Linux придется сосуществовать с Windows.

Можно, конечно, положиться на автоматическую разметку инсталлятора, но она не всегда правильна. Даже если вы устанавливаете Linux на новый жесткий диск, то инсталлятор в большинстве случаев создаст два раздела – один под корневую файловую систему, а другой – для подкачки. Для домашнего использования такая схема вполне имеет право на жизнь. Для сервера – нет. Да и серверы бывают разными.

Если мы создаем FTP-сервер (он же сервер хостер-провайдера) и основное его назначение – хранение данных (например, сайтов) пользователей, то для каталога /home нужно выделить львиную долю дискового пространства. Например, если у вас жесткий диск размером 1 Тб, то для самой системы более чем достаточно 15 Гб дискового пространства (всего 1.5% от емкости жесткого диска), а для каталога /home нужно выделить оставшиеся почти 98% (почти – потому что понадобится еще место под раздел подкачки).

Если же у нас – корпоративный почтовик или сервер баз данных или элементарно прокси-сервер (или же корпоративный сервер, который сочетает в себе все эти функции), тогда основное дисковое пространство нужно выделить под точку монтирования /var. Именно в каталоге /var хранятся файлы базы данных, кэш прокси-сервера Squid, почтовые ящики и т.д.

Недостаток автоматической разметки диска в том, что она не предполагает установку назначения компьютера. Если можно было бы выбрать назначение компьютера, а уже потом выполнять саму разметку, все было бы совсем иначе.

Примечание для домашних пользователей. Если вы производите установку Linux на компьютер с уже установленной Windows, обязательно выполните резервное копирование всех важных данных! Иначе знакомство с Linux начнется для вас с потери данных. Linux нельзя установить на имеющийся раздел. Нужно сначала уменьшить размер этого раздела, а затем на освободившемся месте создать Linux-разделы. Для изменения размера раздела рекомендуется использовать сторонние приложения вроде Acronis, а не штатный инсталлятор Linux – чтобы не было больно за потерянные данные.

1.5.2. Введение в точку монтирования

Точка монтирования – это каталог корневой системы, через который осуществляется доступ к тому или иному разделу. По сути, можно раздел диска можно подмонтировать к любому из каталогов, но обычно используются определенные каталоги, которые имеют определенный смысл для системы, а именно:

- / – корневая файловая система, именно к подкаталогам этой файловой системы и осуществляется монтирование.
- /home – здесь хранятся пользовательские файлы.
- /mnt – обычно этот каталог используется для монтирования с целью доступа к данным, находящимся на другом разделе.
- /usr – сюда обычно устанавливается программное обеспечение (за исключением системного набора программ, который устанавливается в каталоги /bin и /sbin)
- /tmp – каталог для временных файлов.
- /var – здесь хранится переменная информация. К такой относят базы данных, почту, журналы и т.д.

Подробно о назначении каталогов и монтировании мы поговорим в других главах. А пока вам нужно знать следующее. Можно установить Linux на один большой раздел. Скажем, у вас есть жесткий диск (имя устройства /dev/sda), вы на нем создаете всего два раздела – /dev/sda1 и /dev/sda2. Первый будет занимать большую часть дискового пространства, а второй будет размером несколько гигабайтов и будет использоваться для подкачки (подробнее см. след. раздел). Конечно, в современных системах обойтись двумя разделами не получится, но суть остается та же – один для всего, один – для подкачки. Также понадобятся два системных раздела, которые при ручной разметке диска вам придется создать самостоятельно или положиться на автоматическую разметку.

В этом случае все эти каталоги (/home, /usr, /var) будут находиться на одном разделе/диске. С одной стороны ничего страшного, но правильнее будет разместить хотя бы каталоги /home и /var на отдельных разделах, а еще лучше – на отдельных дисках или же организовать RAID-массив. Эти каталоги со-

держат самое ценное – пользовательские данные, поэтому если они будут все находиться на одном жестком диске и он выйдет со строя, приятного будет мало.

Идеально, иметь три жестких диска. На первом жестком диске будет сама система и подкачка, на втором – пользовательские данные (каталог /home), на третьем – каталог /var. Однако, учитывая стоимость жестких дисков, такое расточительство можно себе позволить только, если есть прямая необходимость.

Даже если жесткий диск один, все равно имеет смысл создавать отдельные разделы под системные точки монтирования. Ведь это позволяет на каждом разделе использовать свою файловую систему. Для корневой файловой системы, например, можно использовать файловую систему ReiserFS. Изюминка этой файловой системы в том, что в одном блоке может быть несколько небольших файлов. Например, если размер блока равен 4 Кб, то в нем может поместиться 4 файла по 1 Кб или 2 файла по 2 Кб. Такая файловая система идеально подходит для корневой /, где много файлов конфигурации – все они текстовые и многие из них имеют небольшой размер. Так дисковое пространство будет расходоваться экономнее.

А вот для каталога /var, где нужна высокая производительность, лучше использовать XFS. Это высокопроизводительная файловая система, рассчитанная на большие носители и большие размеры файлов. В общем, если надумаете развернуть Oracle-сервер – это правильный выбор.

Все сказанное ранее – хорошо для сервера. Для домашнего компьютера или для рабочей станции можно использовать два раздела – один для корневой файловой системы (которая будет содержать и данные, и программы), а второй – для подкачки.

1.5.3. Раздел подкачки

В отличие от Windows, Linux использует не файлы, а разделы подкачки. В принципе, если размера раздела подкачки будет недостаточно, то можно создать и файл подкачки, но правильнее создавать именно раздел – так производительность будет выше.

Теперь о размере раздела подкачки. Если оперативной памяти достаточно (например, 8 Гб или больше), тогда раздела подкачки в 8 Гб будет вполне достаточно (размер раздела подкачки равен размеру ОЗУ).

Если оперативной памяти мало, например, 4 Гб или меньше, тогда размер раздела подкачки должен в 2 раза превышать размер оперативной памяти, то есть 8 Гб будет достаточно. Увеличивать раздел подкачки больше не имеет смысла – ведь производительность от этого выше не станет. Назначение раздела подкачки – продолжение работы системы любой ценой. Когда заканчивается оперативная память, сервер должен продолжить работу, хоть и за счет потери производительности (жестким дискам далеко до производительности оперативной памяти). При небольшом объеме ОЗУ лучший тюнинг производительности – дополнительный модуль оперативной памяти.

Когда оперативной памяти много, иногда возникает соблазн вообще отказаться от раздела подкачки. Этого не нужно делать. Если оперативной памяти не хватит, возможны неприятные ситуации. При современных объемах жесткого диска потеря 8-16 Гб дискового пространства (под раздел подкачки) никак не отразится на работе всего сервера. Конечно, как уже было отмечено, в процессе работы системы можно создать и файл подкачки, но его производительность, как показывает практика, ниже, чем производительность раздела подкачки.

1.5.4. Как правильно разбивать жесткий диск?

Правильная разметка жесткого диска выглядит так:

- Для самой системы (точка монтирования /) будет достаточно всего 15 Гб дискового пространства, если не хотите жадничать, можно выделить 20 Гб, но не более.
- Размер раздела подкачки должен быть равен размеру оперативной памяти или превышать его в два раза.
- Разделы должны быть созданы, исходя из выполняемых сервером функций. Об этом мы уже говорили.

Неправильная разметка чревата тем, что придется изменять конфигурацию диска уже в процессе работы сервера, а это чревато только одним – простоем. Потом вам придется выбирать – или не спать какую-то ночь или же получать недовольные звонки от пользователей и/или начальства – если придется перенастраивать сервер в рабочее время.

1.5.5. Ручная разметка в Ubuntu

Если вы не планируете использовать целевой компьютер для других операционных систем, можете выбрать вариант **Стереть диск и установить Ubuntu** (рис. 1.9). Идеальный выбор для нового домашнего/персонального компьютера.

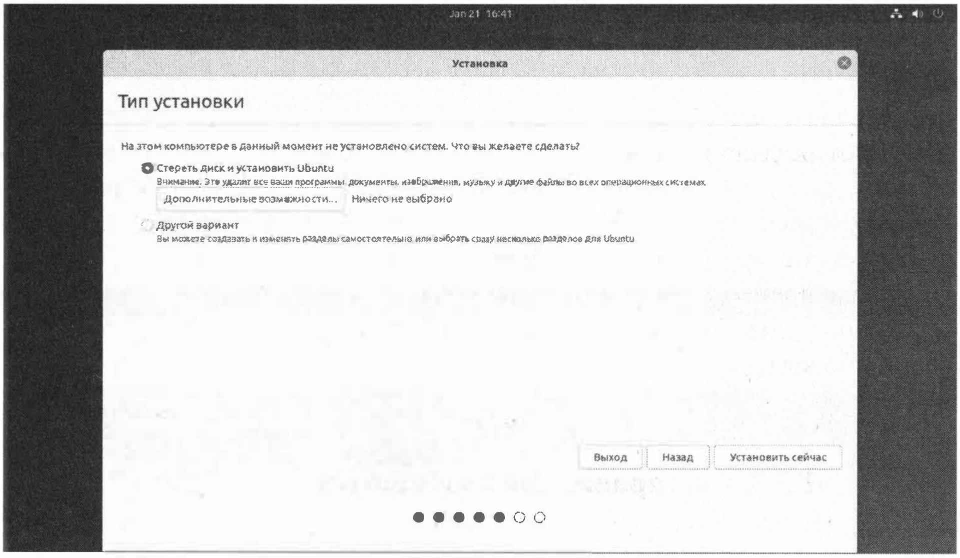


Рис. 1.9. Тип установки

Если же Linux будет использоваться вместе с другой операционной системой или же вы производите настройку сервера, где нужно создать отдельные разделы для разных точек монтирования, нужно выбрать **Другой вариант**.

Инсталлятор Ubuntu довольно удобен для ручной разметки диска. На рис. 1.10 показано, что жесткий диск абсолютно новый и на нем еще не была создана таблица разделов. Для ее создания нажмите кнопку **Новая таблица разделов** (если она уже создана, то кнопка не будет активной). В появившемся окне нажмите кнопку **Продолжить** для подтверждения.

После этого вы увидите, что у вас появилось свободное место (рис. 1.11). Нажмите кнопку + для создания нового раздела. Введите размер создаваемого раздела, выберите точку монтирования и файловую систему (рис. 1.12).



Рис. 1.10. Начало ручной разметки диска

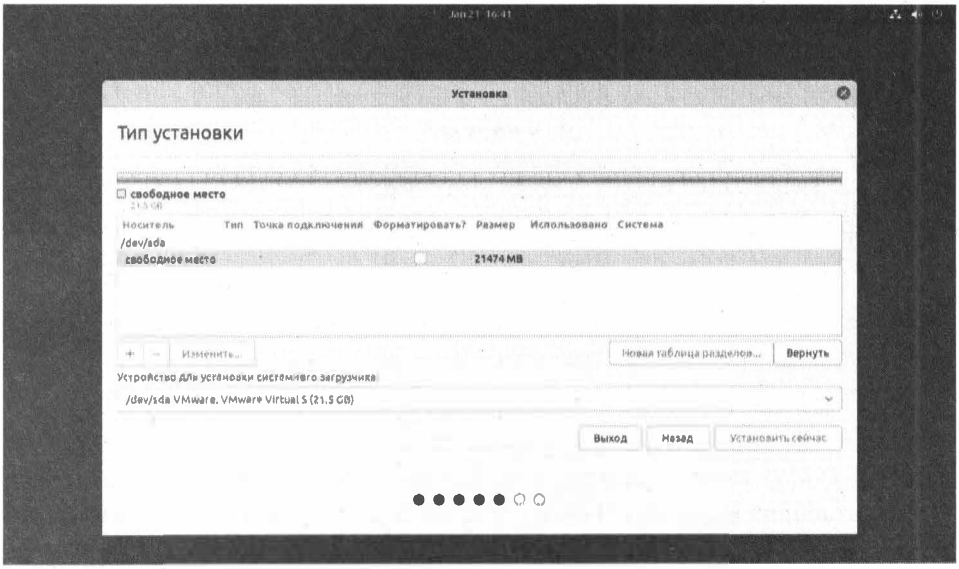


Рис. 1.11. После создания таблицы разделов

Здесь мы создаем раздел размером около 19 Гб для точки монтирования /, файловая система – ext4. На этом разделе будет содержаться и сама система, и данные пользователя.

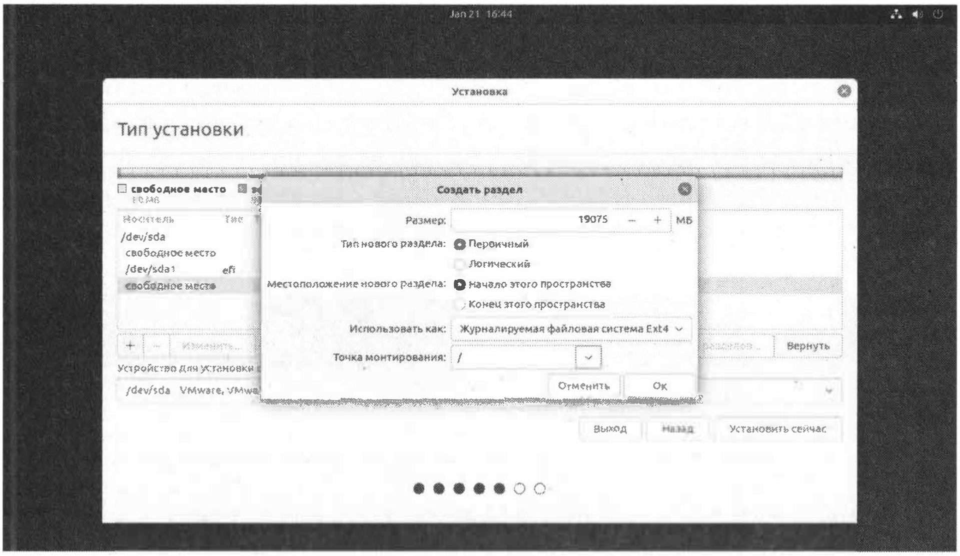


Рис. 1.12. Создание нового раздела

Затем снова щелкните на свободном пространстве и снова нажмите +. Теперь нужно создать раздел подкачки, как показано на рис. 1.13.

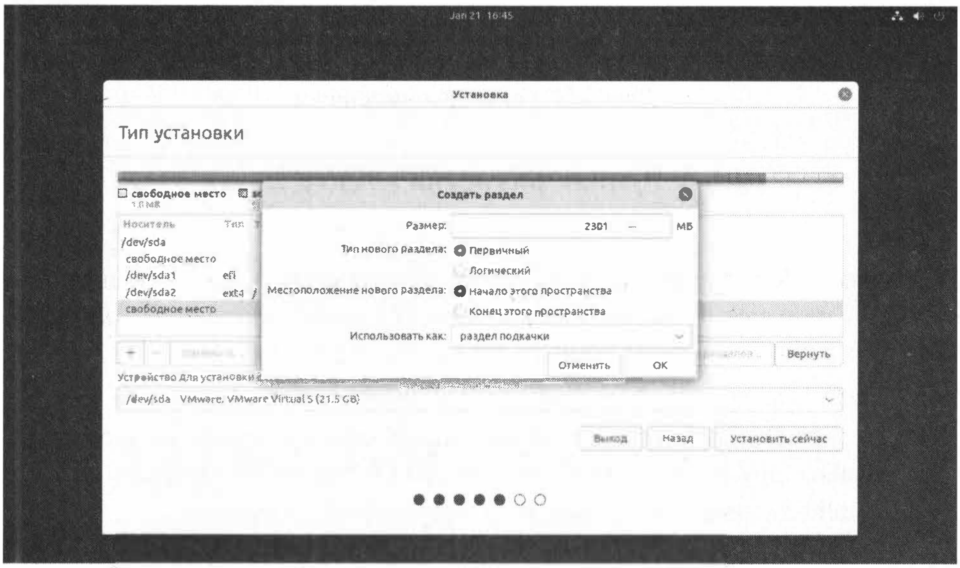


Рис. 1.13. Создание раздела подкачки

Также вам понадобится создать два служебных раздела – **uefi** и **biosgrub** (резервная загрузочная область BIOS). Считается, что для первого будет достаточно 100 Мб, для второго – 250. Я экспериментировал с разными параметрами и вполне рабочей оказалась конфигурация, где под первый раздел я выделили всего 9 Мб, под второй – 89, как показано на рис. 1.14. Но все я это делал в виртуальной машине (по другому скриншоты для книги создавать неудобно), а вам же лучше не рисковать и выделить хотя бы по 100 Мб для каждого раздела.

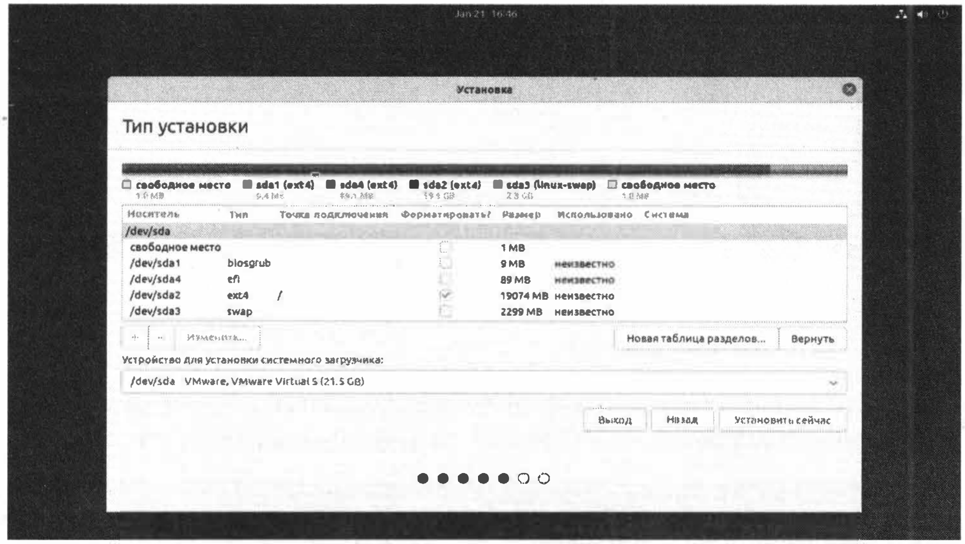


Рис. 1.14. Разметка завершена

1.5.6. Ручная разметка в Astra Linux

Установки в Astra Linux такой же, как в Debian. Debian – неплохой дистрибутив, но у него самый неудобный для разметки диска установщик. Наверное, над ним работали профессионалы по самым неудобным GUI. И у них это получилось!

Хорошо, если вы настраиваете персональный компьютер, на котором не будет никаких других операционных систем. Тогда можно выбрать вариант **Авто – использовать весь диск** и не заморачиваться (рис. 1.15)

В противном случае выбираем **Вручную**. Далее выбираем ваш жесткий диск и нажимаем кнопку **Продолжить**.

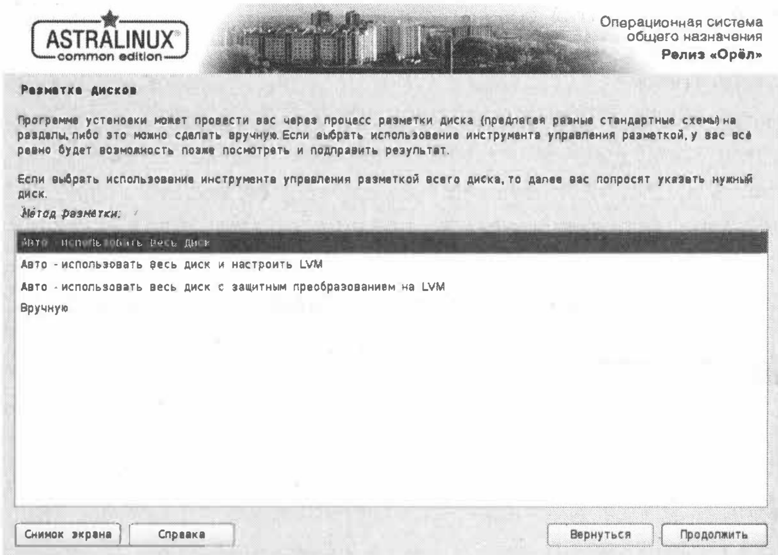


Рис. 1.15. Разметка в Astra Linux

Создайте новую таблицу разделов (рис. 1.16). Затем выделите свободное место (как показано на рис. 1.17) и снова нажмите кнопку **Продолжить**.

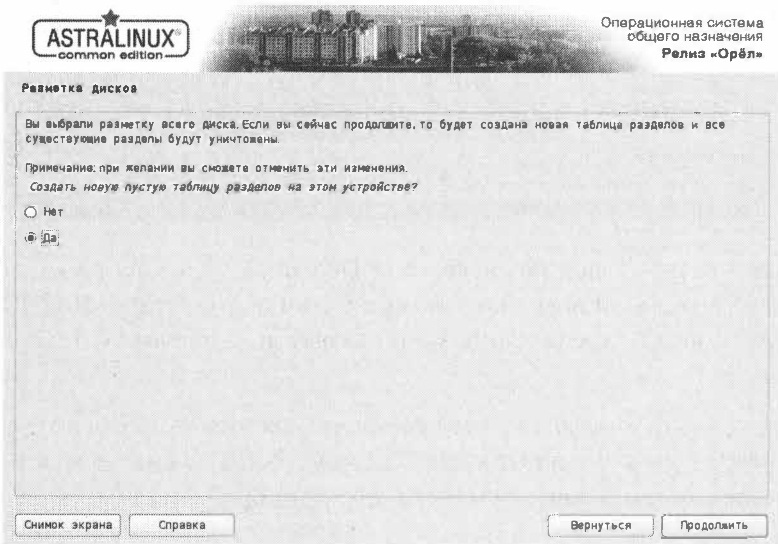


Рис. 1.16. Новая таблица разделов



Рис. 1.17. Выберите свободное место и нажмите кнопку Продолжить

В появившемся меню (рис. 1.18) выберите команду **Создать новый раздел** и снова нажмите сами знаете какую кнопку. Нужно будет ввести размер раздела (рис. 1.19), выбрать его тип (первичный/логический), расположение (начало/конец), далее появится возможность установить другие параметры раздела (рис. 1.20). Вы можете изменить точку монтирования, параметры монтирования, метку и т.д. Параметр **Использовать как** позволяет изменить файловую систему раздела.

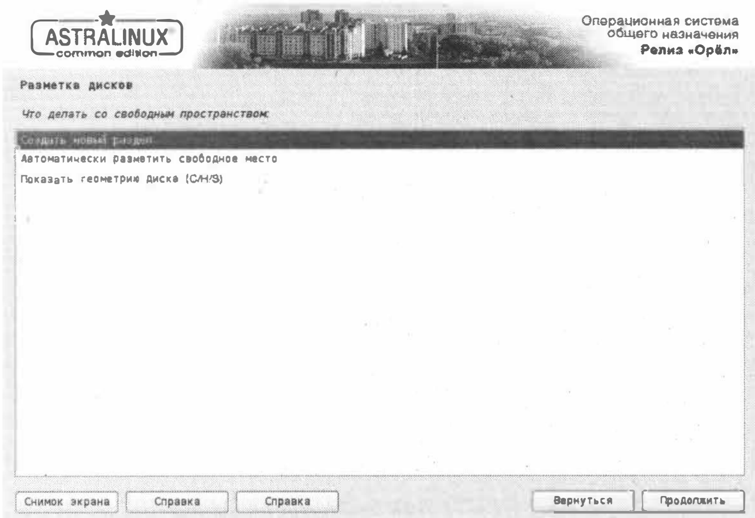


Рис. 1.18. Создание нового раздела



Рис. 1.19. Размер создаваемого раздела



Рис. 1.20. Настройка раздела

Изменение происходит путем выделения нужного параметра и нажатия кнопки **Продолжить**. Когда все будет готово, используйте команду **Настройка раздела закончена**. Аналогичным образом создайте раздел подкачки (рис. 1.21).

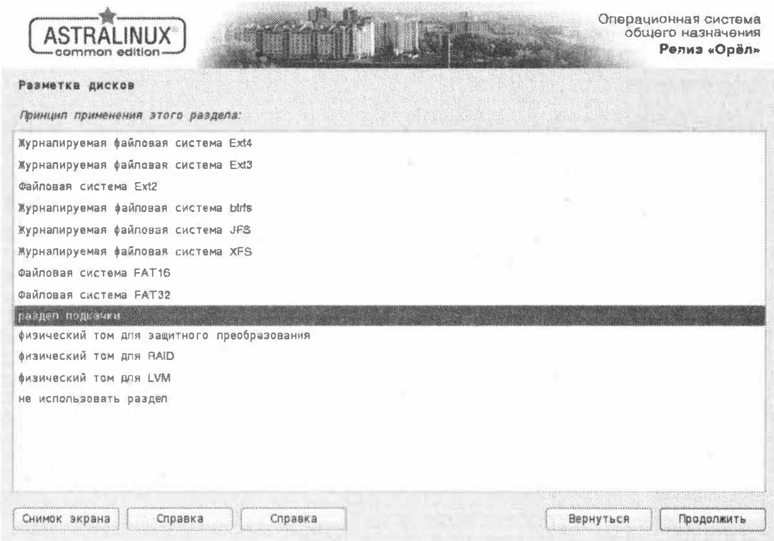


Рис. 1.21. Выбор файловой системы для раздела подкачки

Когда все будет готово, используйте команду **Закончить разметку и записать изменения на диск**. На следующем экране нужно согласиться с форматированием разделов (рис. 1.23). Средство разметки диска очень неудобное, но и к нему можно привыкнуть.



Рис. 1.22. Готовая разметка

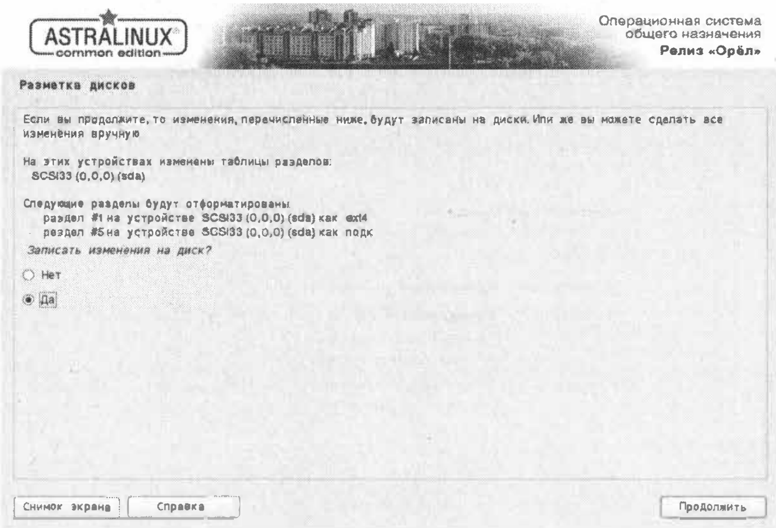


Рис. 1.23. Подтверждаем изменения

1.6. Установка пароля администратора

Раньше при установке Linux предлагалось создать пароль пользователя *root* – пользователя с максимальными правами и создать обычного пользователя. Сейчас же дистрибутивы отошли от использования пользователя *root* по соображениям безопасности, а эта учетная запись часто оказывается заблокированной. Делается это по понятной причине – имя *root* знают все, поэтому злоумышленнику нужно подобрать только один параметр – пароль, а если он не будет знать имя пользователя, то придется угадывать еще и логин, что усложняет задачу.

Современные дистрибутивы предлагают создать так называемого административного пользователя – пользователя, которому разрешено выполнять команду *sudo* для получения максимальных полномочий. Это и безопасно, и удобно для самого пользователя – ему нужно помнить один пароль, а не два (пользователя *root* и обычного пользователя).

При создании административного пользователя (рис. 1.24) вы задаете имя пользователя и его пароль. Ubuntu допускает использование слабого пароля: если пароль является слабым для подбора, инсталлятор просто сообщает об этом. А вот Astra Linux не позволяет использовать пустые пароли – пароль должен быть минимум 8 символов и содержать хотя бы буквы с цифрами.

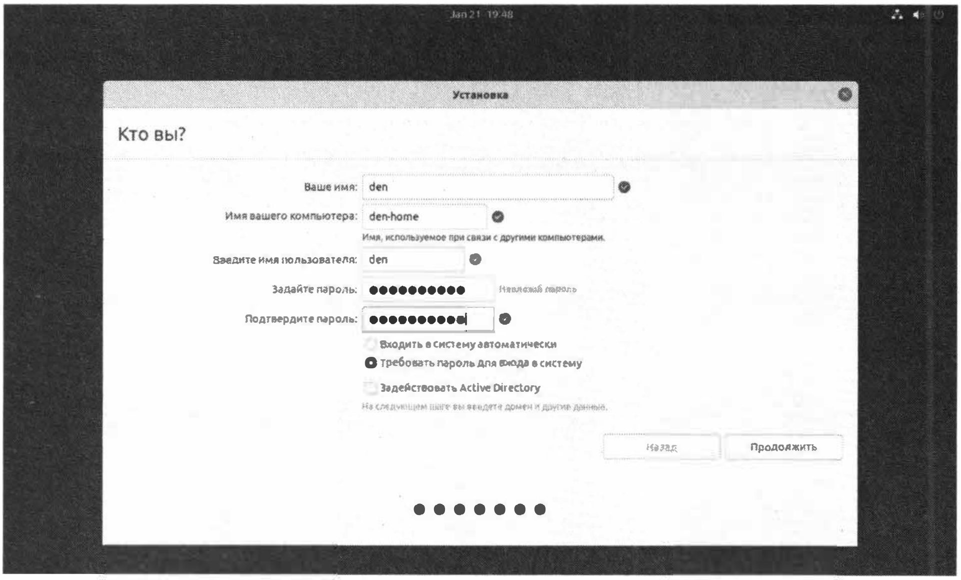


Рис. 1.24. Создание пользователя в Ubuntu



Рис. 1.25. Простые пароли в Astra Linux использовать нельзя

Также обратите внимание на параметр **Входить в систему автоматически** (Ubuntu). Он позволяет не вводить пароль при входе в систему, что хорошо для домашних компьютеров. В Astra Linux подобный параметр доступен после установки системы – он называется **Включить автологин в графическую сессию** (рис. 1.26). В следующей главе будет показано, как включить/выключить автовход уже после установки системы.

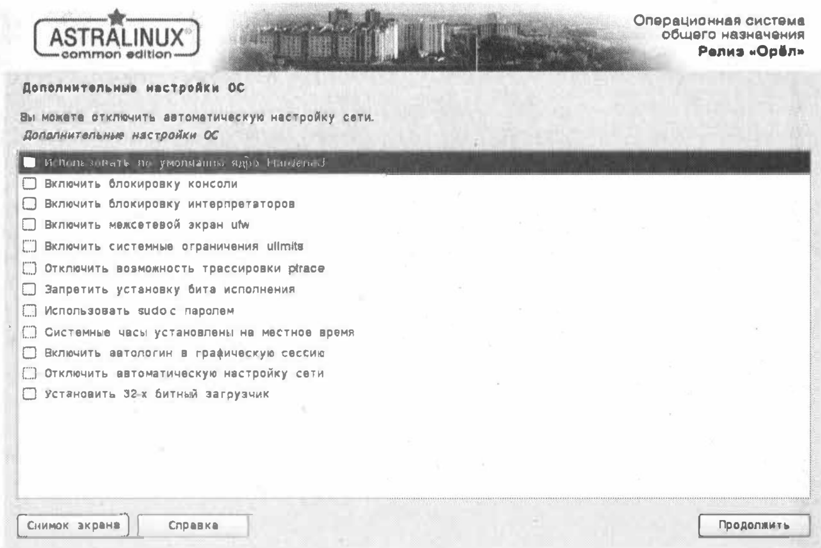


Рис. 1.26. Включение автовхода в Astra Linux

1.7. Параметры программного обеспечения

В Astra Linux позволяют выбрать наборы программного обеспечения (рис. 1.27). В принципе любой пакет можно доустановить после установки системы, но при желании вы можете сразу выбрать нужную группу. Опять-таки если вы заботитесь о дисковом пространстве, лучше не устанавливать готовые наборы, а установить только те пакеты, которые вам нужны. Некоторые, например, Игры можно сразу отключать – вы ними не будете пользоваться.

В Ubuntu ничего такого нет. При установке можно выбрать только обычную или минимальную установку, а также выбрать обновление ПО во время установки и опцию установку стороннего ПО. Рекомендую включить этот переключатель, чтобы сразу у вас заработали мультимедиа-форматы. Кодеки нельзя по условиям лицензии распространять вместе с Linux, но их можно установить после установки системы (бесплатно). Этот переключатель как раз и производит данную установку.

Сразу после установки Ubuntu предложит вам доустановить необходимые программы (рис. 1.28). Вы можете сделать это сразу, а можете по мере необ-

ходимости. Лучше всего нажать кнопку **Открыть Менеджер приложений сейчас** – выбор приложений будет гораздо шире (рис. 1.29).

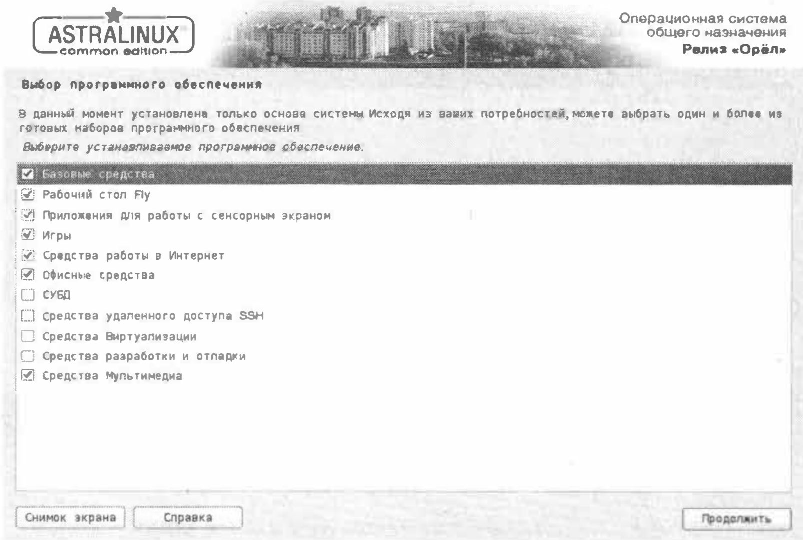


Рис. 1.27. Доступные по умолчанию наборы ПО

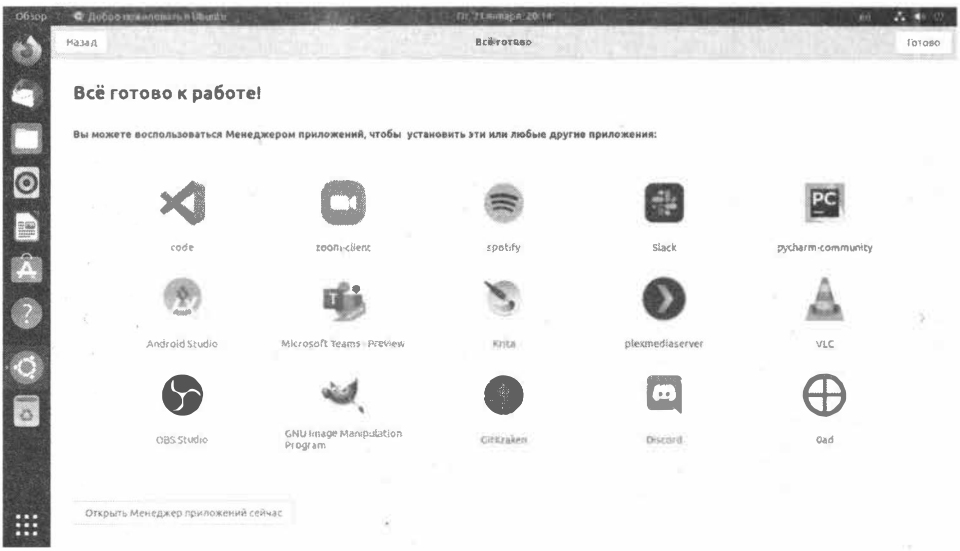


Рис. 1.28. Предложение установить некоторые программы

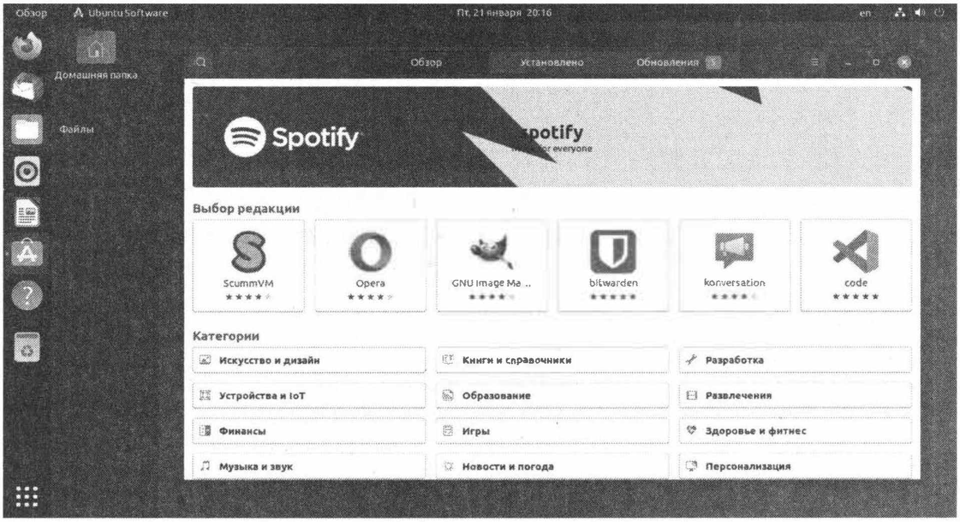


Рис. 1.29. Менеджер приложений Ubuntu

1.8. Установка загрузчика

Ubuntu устанавливается проще. По сути, самое сложное – это разметка диска. Но в Astra Linux нужно еще выбрать, куда устанавливать загрузчик Linux. Обычно его нужно установить в /dev/sda. Но если у вас есть уже другая операционная система и вы хотите загружать Linux не с помощью родного загрузчика GRUB2, а посредством стороннего загрузчика, тогда нужно или установить загрузчик в другое место или же не устанавливать его вовсе.

Мы рассмотрели все основные моменты, на которые нужно обратить внимание при установке Linux.

Далее мы поговорим о входе в систему и о завершении работы.



Рис. 1.30. Выбор устройства для установки загрузчика

Глава 2.

Вход в систему



В этой небольшой главе мы рассмотрим вход в систему, основные элементы графического интерфейса, а также правильное завершение работы в системе.

2.1. Вход в консоль и переключение между ними

Вход в систему в Linux ничем не отличается от входа в систему в любой операционной системе. Нужно ввести имя пользователя и пароль. Вход может быть как в текстовом, так и в графическом режиме. На сервере графический интерфейс, как правило, не устанавливается, чтобы не расходовать драгоценные системные ресурсы на никому ненужный графический интерфейс. На рис. 2.1 изображен вход в Astra Linux. На рис. 2.2 изображен графический вход в Astra Linux.

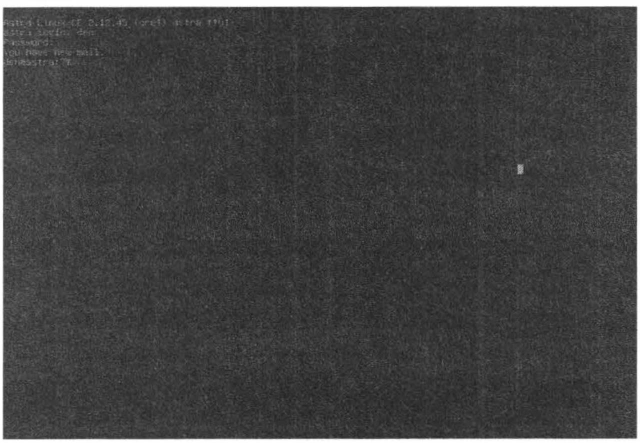


Рис. 2.1. Вход в систему (Astra Linux)

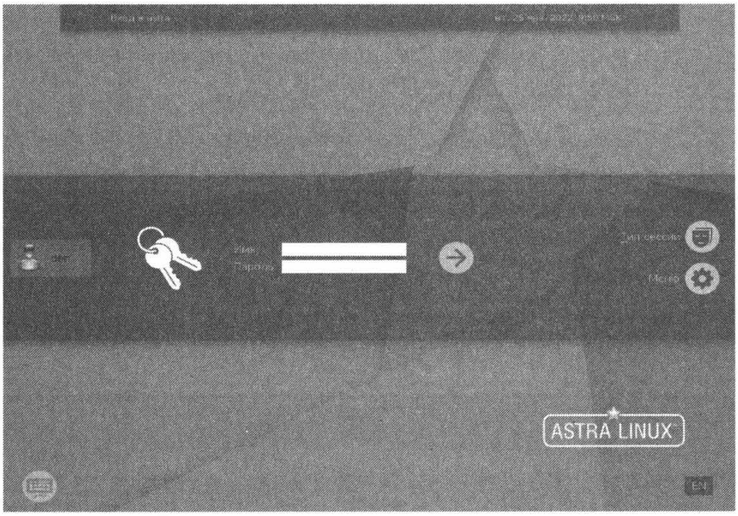


Рис. 2.2. Графический вход (Astra Linux)

Давайте разберемся, что есть что. Самая первая строка (рис. 2.1) – название и версия дистрибутива. Далее выводится имя терминала, на котором вы сейчас находитесь (tty1). Некоторые другие дистрибутивы еще сообщают версию ядра и архитектуру систему. Здесь разработчики посчитали лишним вывод этой информации.

Далее нужно ввести логин (в нашем случае *root*) и пароль. Обратите внимание, что пароль не отображается при вводе, не отображаются даже звездочки. В графическом режиме вместо введенных символов могут отображаться звездочки или точки.

После входа в систему вы увидите или приглашение командной строки или графический интерфейс. В первом случае приглашение будет иметь вид # (если вы вошли как *root*) или $ (если вы вошли, как обычный пользователь).

Перед приглашением командной строки выводится время и дата последнего входа в систему, а также терминал, на котором был осуществлен вход (в нашем случае – tty1). Значение :0 означает, что последний вход был в графическом режиме.

Строка **You have mail** означает, что у вас есть новые сообщения электронной почты. Для их чтения нужно или использовать какую-то почтовую программу, настроенную на чтение системных сообщений e-mail, либо же команду *mail*.

Для переключения между терминалами используются комбинации клавиш Alt + F1 … Alt + F6. Комбинация клавиш Alt + F7 переключит вас из консоли в графический режим, если система X11 запущена.

Находясь в графическом режиме, вы можете нажать комбинацию клавиш Ctrl + Alt + Fn, где *n* – это число от 1 до 6 для переключения на нужную вам консоль. Функционал экрана входа в систему отличается в зависимости от дистрибутива – здесь все на усмотрение разработчиков. В Ubuntu он минималистичен – выводятся только имена пользователей, после выбора пользователя можно будет ввести пароль и войти в систему. Из полезных "фич" – только смена раскладки клавиатуры (рис. 2.3).

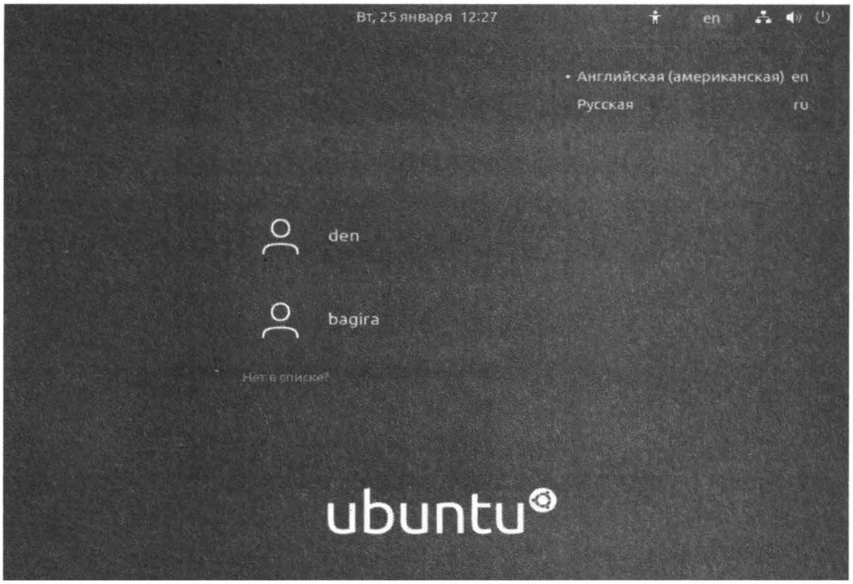


Рис. 2.3. Вход в систему (Ubuntu 21.10)

В Astra Linux можно выбрать тип сессии (рис. 2.4), а также открыть меню (рис. 2.5), в котором будут полезные команды, такие как переключение на консоль (на случай, если вы не знаете ничего о комбинации Ctrl + Alt + F1), вызов экранной клавиатуры, перезапуск X-сервера и команда выключения системы.

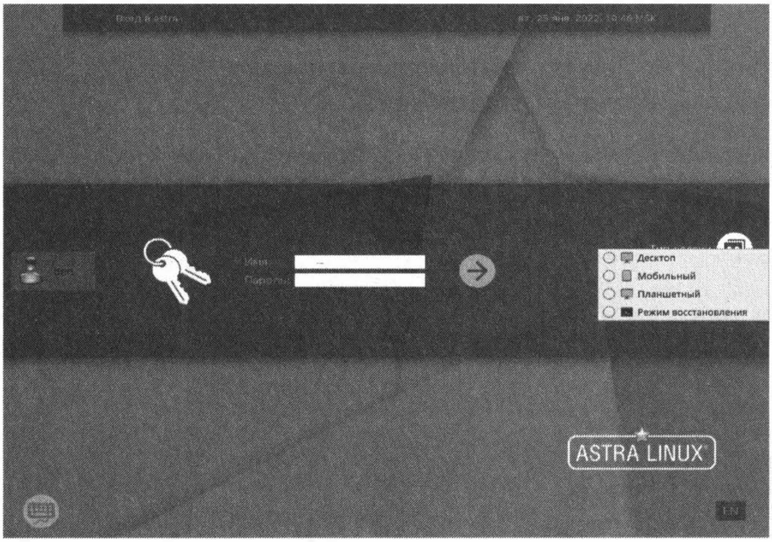


Рис. 2.4. Выбор типа сеанса (Astra Linux)

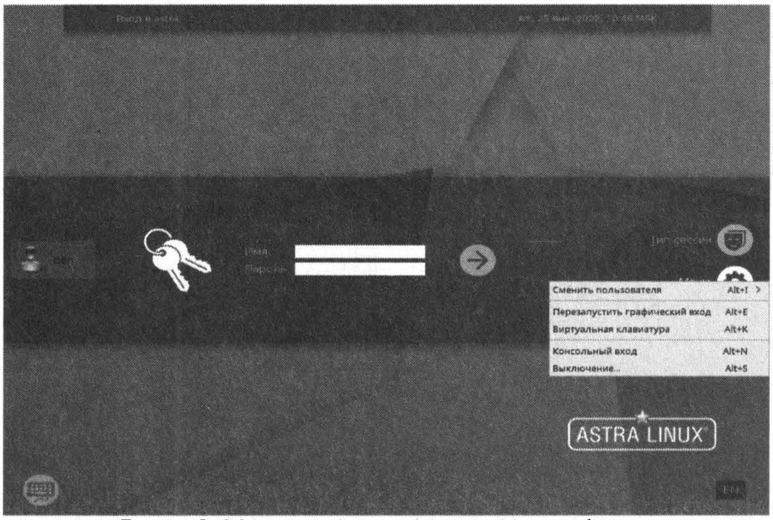


Рис. 2.5. Меню входа в систему (Astra Linux)

2.2. Основные элементы графического интерфейса

Вы вошли в систему, но что делать дальше? Для бывшего Windows-пользователя слегка непривычно. Далее мы рассмотрим основные элементы интерфейса пользователя Ubuntu и Astra Linux.

2.2.1. Интерфейс Ubuntu

По умолчанию в Ubuntu установлена графическая среда GNONE 2.36. Интерфейс минималистичен. Ничего лишнего.

Слева выводится панель, на которую можно поместить кнопки вызова часто используемых приложений (рис. 2.6). В самом низу этой панели есть кнопка, позволяющая открыть экран **Приложения** (рис. 2.7).

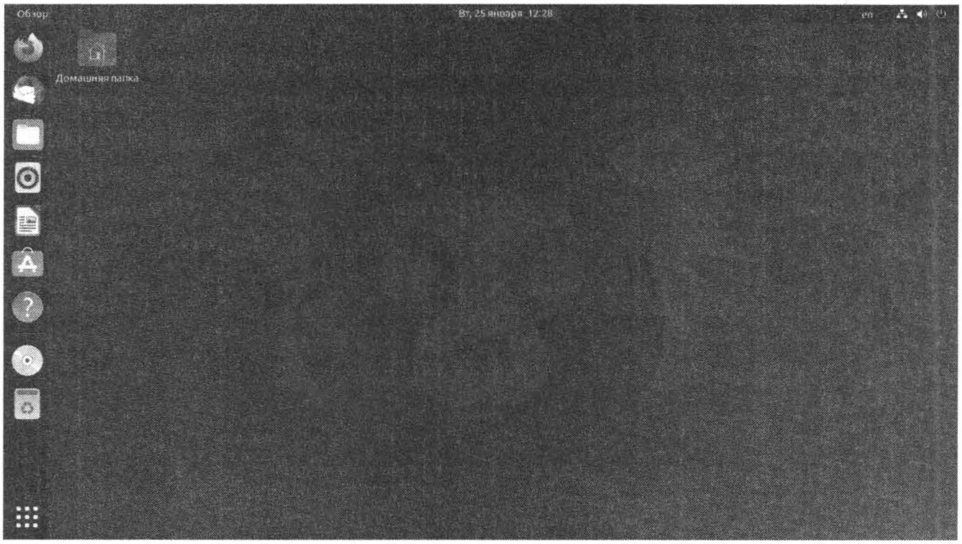


Рис. 2.6. Рабочий стол Ubuntu 21.10

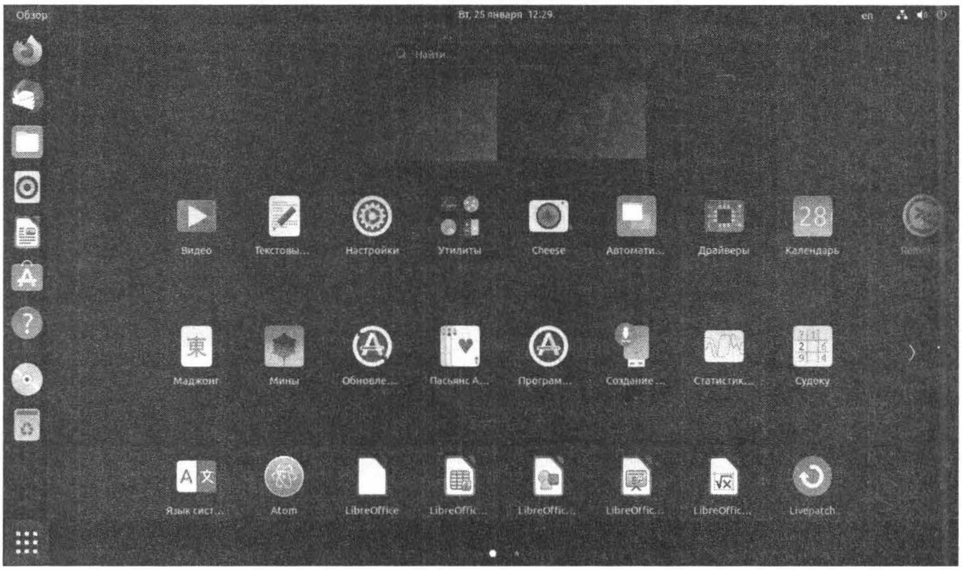


Рис. 2.7. Приложения

Обратите внимание: по умолчанию этот экран отображает только популярные приложения (те, которые вы чаще всего использовали) и может показаться, что приложений очень мало. На самом деле это не так. Перейдите на вкладку **Все** (внизу экрана) и вы увидите все установленные приложения. Поле **Найти** вверху экрана позволяет быстро найти то или иное приложений.

Чтобы добавить значок приложения на панель слева, просто перетащите его в нужную позицию или же щелкните по значку правой кнопкой мыши и выберите команду **Добавить в избранное**. Аналогично, команда **Удалить из избранного** удаляет этот значок с панели избранного.

По умолчанию на панели избранного находятся следующие приложения (сверху вниз):

- Браузер Firefox
- Почтовый клиент Thunderbird
- Файловый менеджер
- Музыкальный проигрыватель
- Текстовый процессор Writer
- Центр установки ПО Ubuntu Software
- Кнопка вызова справочной системы
- Кнопка быстрого доступа к DVD

Когда вы запускаете приложение, его значок также добавляется на панель избранного. Другими словами, она выполняет роль еще и панели задач, которую вы можете использовать для переключения между приложениями. Запущенные приложения отмечаются кружочком слева от значка кнопки. На рис. 2.8 показано, что запущены браузер, файловый менеджер и терминал (средство для ввода команд).

Существует несколько способов переключения между запущенными приложениями:

1. Нажатие на кнопку приложения на панели избранного
2. Комбинация клавиш Alt + Tab, выводящая все запущенные приложения (как в Windows)
3. Кнопка **Обзор** в верхнем левом углу экрана (рис. 2.9)

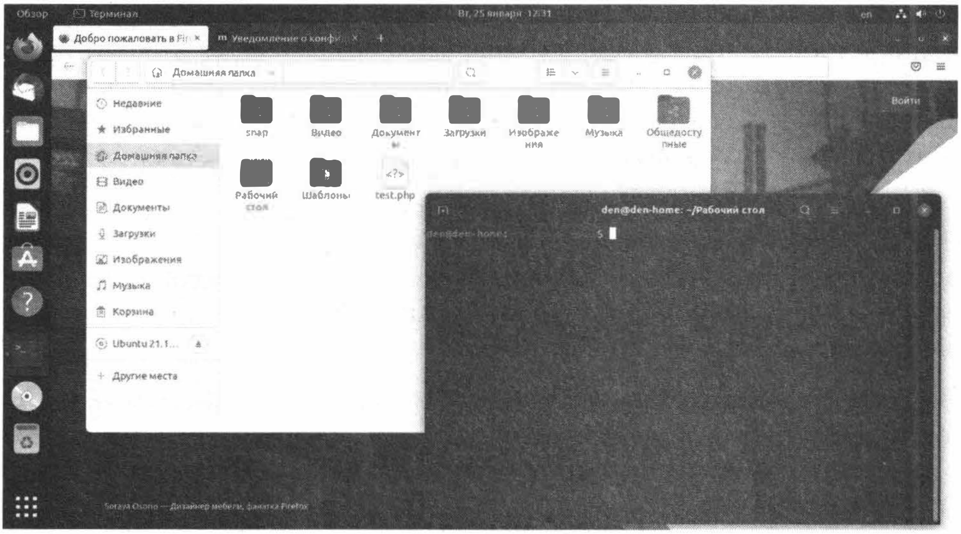


Рис. 2.8. Демонстрация запущенных приложений

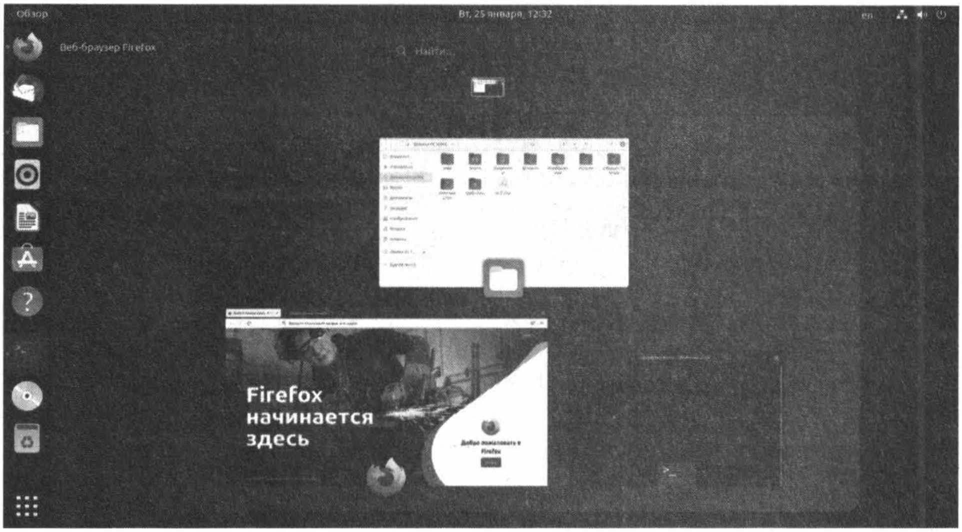


Рис. 2.9. Экран Обзор

В верхнем правом углу экрана находятся довольно важные элементы графического интерфейса: переключатель раскладки клавиатуры и меню, позволяющее управлять сетью, звуком и завершением работы. Нажмите на него (рис. 2.10). Вы увидите ползунок, позволяющий регулировать громкость, далее надпись "Проводное сеть подключена" означает, что на данный

момент компьютер подключен к проводной локальной сети, нажатие по этой надписи позволит управлять другими сетевыми соединениями. Кнопка **Настройки** вызывает средство настройки графического интерфейса. Кнопка **Заблокировать** позволяет заблокировать экран пользователя – полезно, если вы хотите ненадолго отойти. Кнопка **Выключить / Завершить сеанс** открывает меню завершения работы:

- **Завершить сеанс** – производит выход пользователя из системы, но не выключает компьютер. Если компьютер используется не только вами, данная команда используется, когда вы закончили работу и готовы передать компьютер другому пользователю.
- **Сменить пользователя** – позволяет войти под именем другого пользователя, но не завершать сеанс текущего пользователя.
- **Ждущий режим** – переводит компьютер в ждущий режим для экономии электричества. Компьютер не завершает работу, приложения не закрываются, но отключать его от питания нельзя, иначе все несохраненные изменения будут потеряны.
- **Выключение** – вызывает дополнительное меню, в котором вы сможете или выключить компьютер или перезагрузить его.

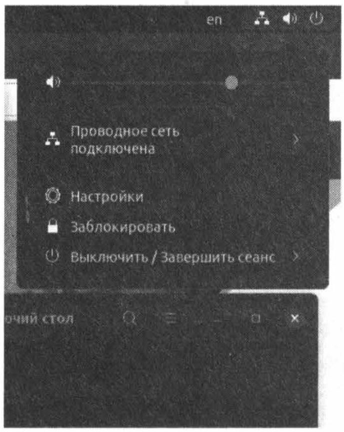


Рис. 2.10. Системное меню

Выберите команду **Настройки**. Сейчас "пройдемся" по наиболее полезным настройкам графического интерфейса. Настроек очень много, учитывая, что интерфейс достаточно качественно переведен на русский язык, не составит особого труда разобраться со всеми настройками самостоятельно. Но на некоторые следует обратить внимание.

В разделе **Фон** можно изменить фон рабочего стола (рис. 2.11). Кнопка **Добавить изображение** позволяет добавить в галерею пользовательское изображение, если стандартные вам не нравятся.

> **Примечание**. Много хороших (и главное бесплатных) обоев можно найти по адресу https://unsplash.com/wallpapers. На этом сайте вы найдете обои на любой вкус.

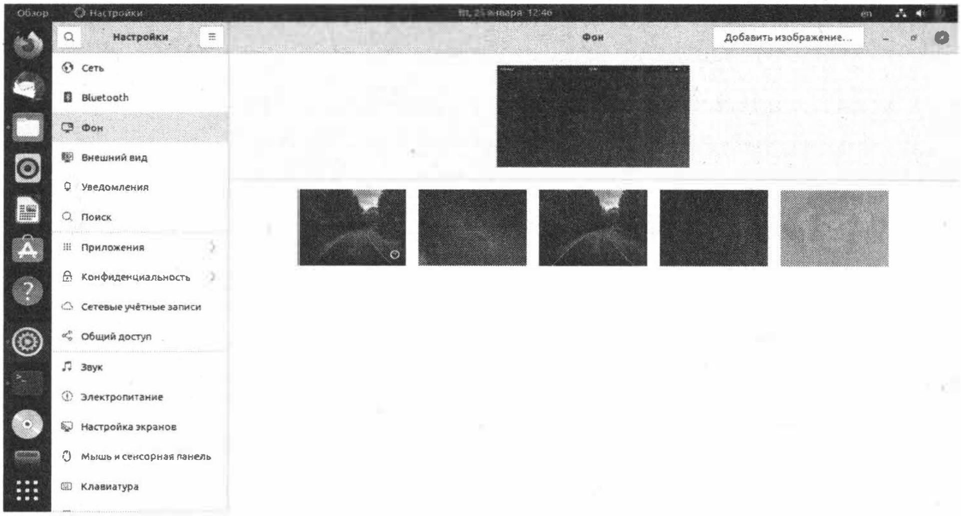


Рис. 2.11. Раздел Фон

Раздел **Внешний вид** позволяет выбрать модную нынче темную тему оформления. Нужно отметить, что темная тема впервые появилась в Ubuntu 20.04, так что на сегодняшний день – это новинка. Также здесь можно выбрать расположение панели задач. Уверен, что пользователи Windows выберут положение **Снизу**, чтобы было привычнее.

В разделе **Конфиденциальность, Блокировка экрана** можно настроить или вовсе отключить блокировку экрана. Задержка выключения экрана в 5 минут просто раздражает – отвлекся, и экран уже потух и заблокировался, после чего нужно вводить пароль заново. Поэтому, если вы не страдаете паранойей, задержку выключения экрана можно установить в 15 минут, остальные параметры оставьте без изменения. Так будет комфортнее.

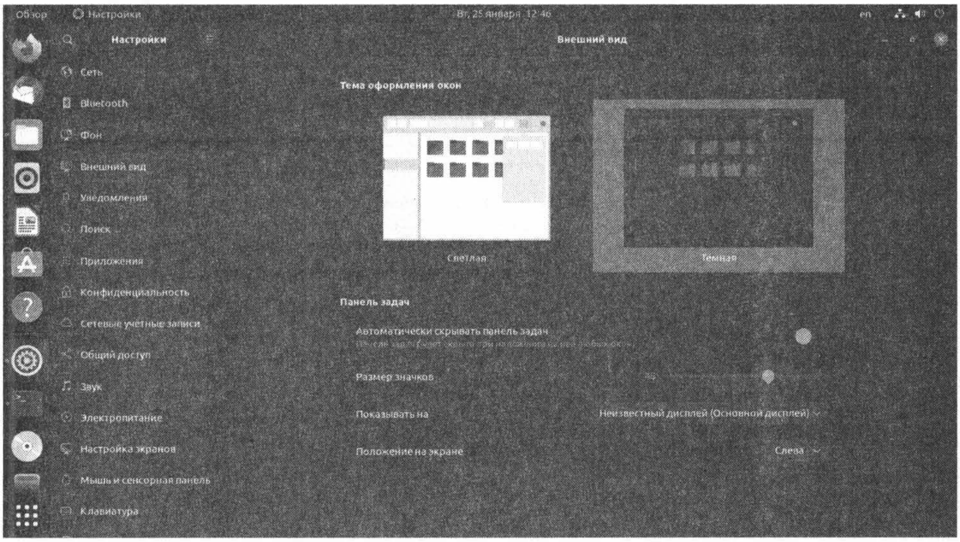


Рис. 2.12. Раздел Внешний вид. Темная тема

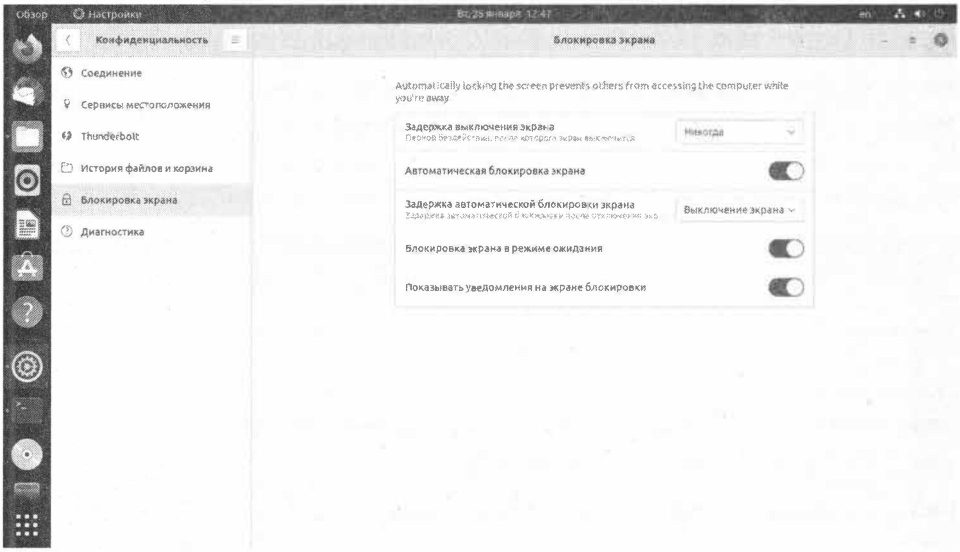


Рис. 2.13. Раздел Блокировка экрана

В разделе **Конфиденциальность** также находится раздел **История файлов и корзина**. Ubuntu хранит историю использования файлов. Здесь можно задать продолжительность хранения истории и параметры очистки. Также здесь можно очистить корзину и временные файлы, а также задать интервал автоматического удаления, который по умолчанию равен 30 дней.

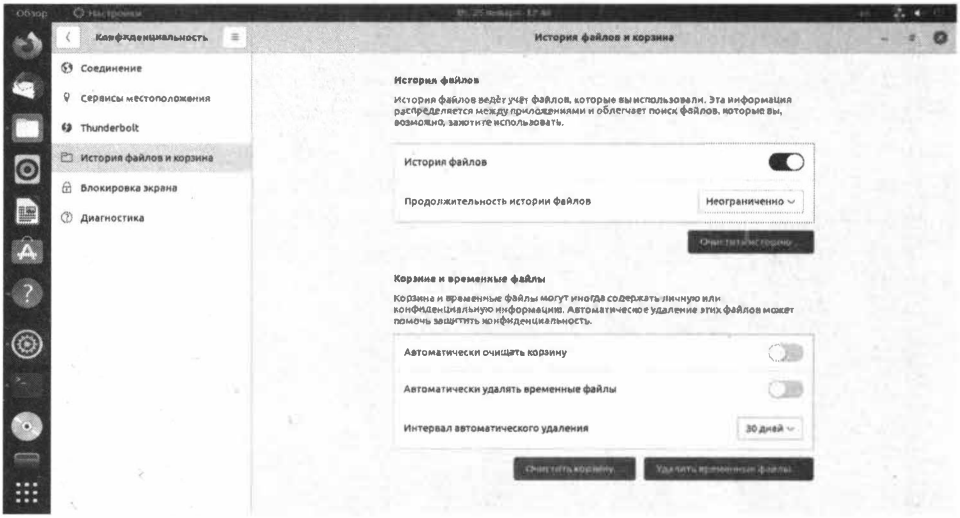


Рис. 2.14. История файлов и корзина

Раздел **Настройка экранов** позволяет выбрать разрешение экрана монитора, если разрешение по умолчанию выбрано неправильно (рис. 2.15).

Раздел **Комбинации клавиш** позволяет изменить комбинации клавиш, в том числе для переключения раскладки клавиатуры, а в разделе **Дата и время** можно выбрать часовой пояс, установить дату и время. Разделы **Сеть** и **Пользователи** будут рассмотрены в других главах этой книги.



Рис. 2.15. Настройка экранов

2.2.2. Интерфейс Astra Linux

Интерфейс Astra Linux пытались максимально заточить под Windows. Панель задач снизу, меню в стиле **Пуск**, только открывается окно не кнопкой с логотипом Windows, а кнопкой с лого Astra (рис. 2.16). Бывшие Windows-пользователи оценят такое решение – для них такой интерфейс будет привычнее интерфейса Ubuntu.

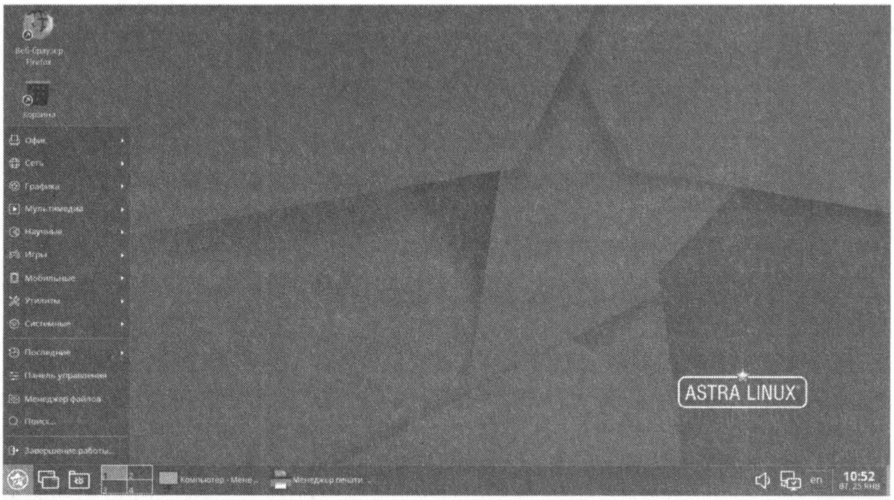


Рис. 2.16. Рабочий стол и главное меню Astra Linux

Рассмотрим содержимое панели задач. Первая кнопка открывает главное меню, изображенное на рис. 2.16. Вторая кнопка используется для переключения между окнами – когда вам нужно быстро найти запущенное окно (рис. 2.17).

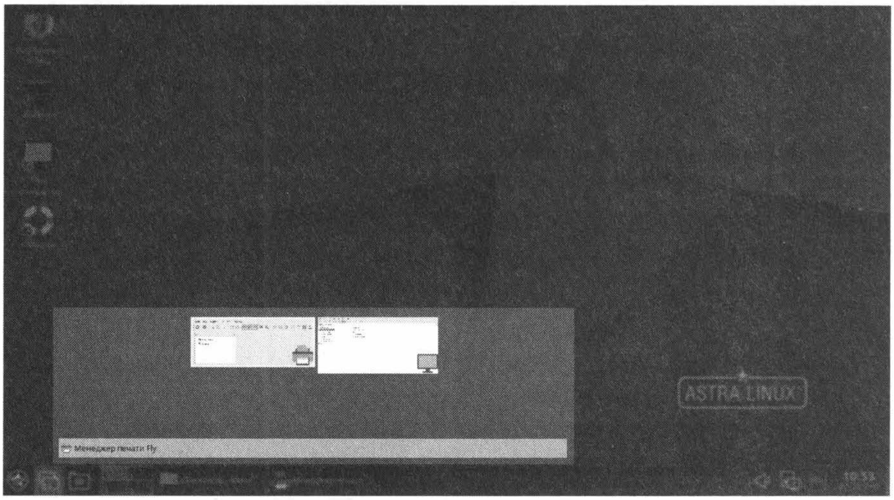


Рис. 2.17. Переключение между окнами

Третья кнопка открывает файловый менеджер (рис. 2.18). Далее следует переключатель рабочих столов. Каждый рабочий стол – это отдельное пространство, на котором вы можете запускать приложения. Рабочие столы позволяют эффективно организовать рабочее пространство, когда запущенных окон много. Для перемещения уже открытого окна на другой рабочий стол щелкните правой кнопкой мыши по заголовку окна и выберите меню Рабочий стол, далее выберите номер рабочего стола (рис. 2.19).

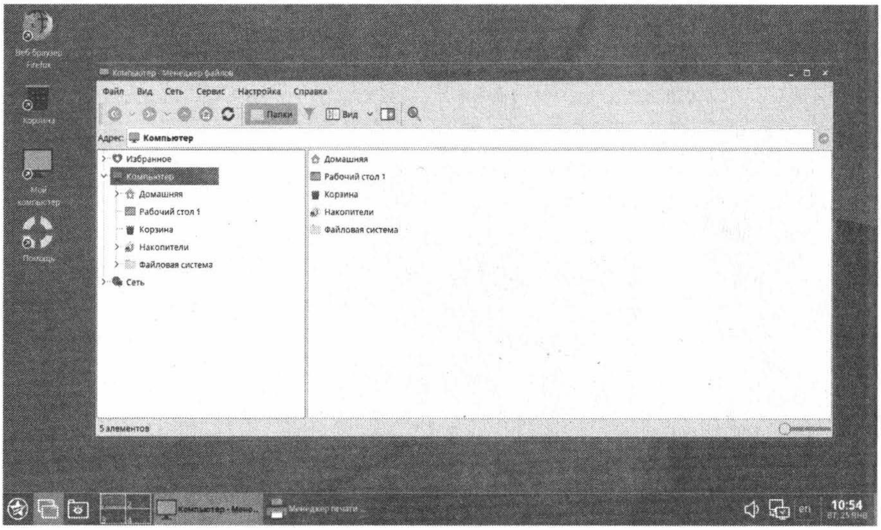


Рис. 2.18. Файловый менеджер

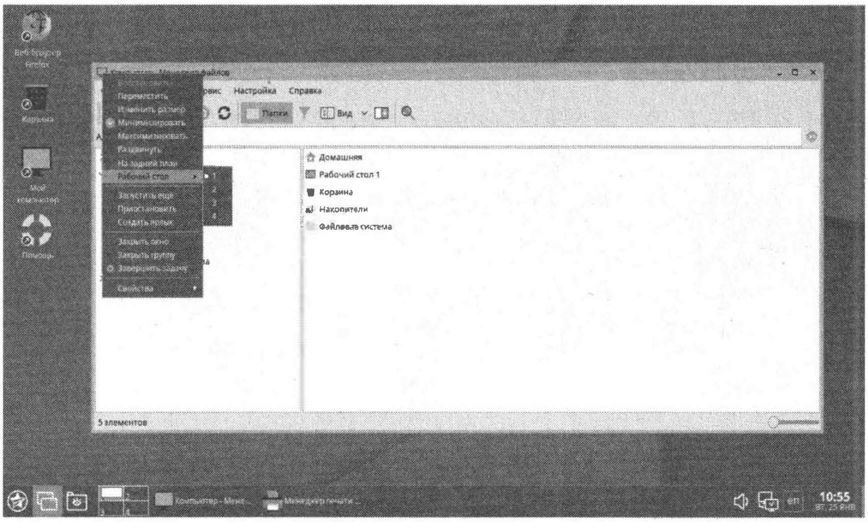


Рис. 2.19. Перемещение окна на другой рабочий стол

В Ubuntu также можно организовать работу нескольких рабочих столов, но в последних версиях Ubuntu стараются использовать концепцию одного рабочего стола – минимизация интерфейса.

Снизу справа отображаются системные значки – менеджера сети, средства проверки обновлений, менеджера внешних устройств хранения, регулятора громкости, переключателя раскладки клавиатуры и средства вывода даты и времени. Все эти значки – интерактивные, то есть вы можете щелкнуть по значку *менеджера сети*, чтобы отключиться от текущего соединения. Если щелкнуть по нему правой кнопкой мыши, появится команда **Изменить соединения** – для управления соединения. Нам понравилась команда **Включить поддержку сети**, которая включает/выключает поддержку сети. Она может понадобиться, если Wi-Fi соединение неожиданно "отвалится" – не перезагружать же компьютер из-за этого?

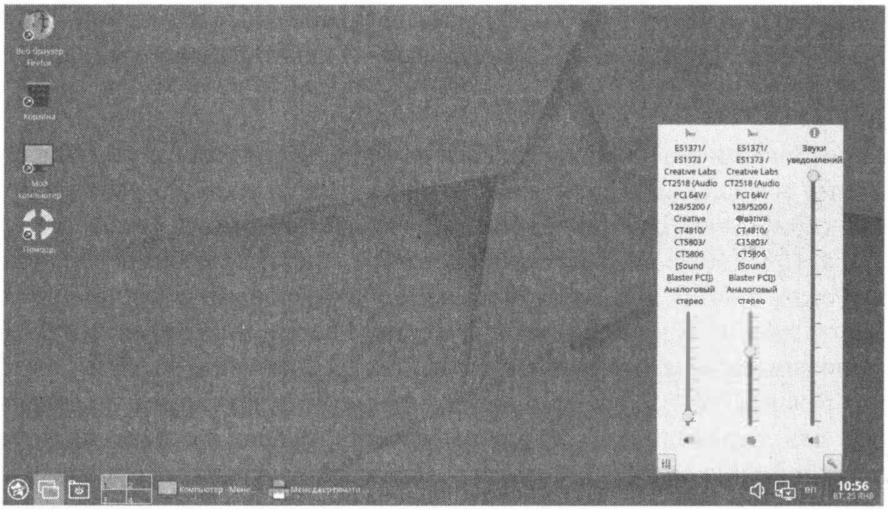


Рис. 2.20. Регулятор громкости

Щелчок по *регулятору громкости* открывает микшер громкости, позволяющий регулировать громкость для динамиков, микрофонов и отдельный регулятор предусмотрен для звуков уведомлений (рис. 2.20).

Много полезных утилит вы найдете в программных группах **Утилиты** и **Системные**. Так, здесь вы найдете:

- **Терминал Fly** – средство для ввода команд Linux.

- **Политика безопасности** – здесь можно добавлять других пользователей Linux и изменять пароль уже имеющихся.
- **Менеджеров пакетов Synaptic** – используется для установки программного устройства. Довольно удобный менеджер и жаль, что в последних версиях Ubuntu от него отказались.
- **Менеджер устройств** – позволяет просматривать имеющиеся в системе устройства.
- **Редактор разделов Gparted** – графический редактор разметки, который может использоваться для разметки жесткого диска, например, при подключении нового жесткого диска.
- **Запись ISO образа на USB носитель** – название этой утилиты вполне описывает ее назначение.
- **Менеджер файлов MC** – запускает двухпанельный менеджер файлов Midnight Commander, очень популярный у Linux-пользователей.

Команда **Завершение работы** открывает меню, в котором будут различные варианты завершения работы – выключение, перезагрузка, сон, блокировка, выход, гибернация. Последний режим позволяет сохранить состояние компьютера. При следующей загрузке состояние компьютера будет восстановлено. Режим гибернации похож на режим сна, но поскольку состояние оперативной памяти будет сохранено на жестком диске, вы можете выключить питание компьютера, что никак не повлияет на состояние системы, в отличие от режима сна. Не все компьютеры поддерживают режим гибернации, также для перехода в этот режим нужно, чтобы на жестком диске было достаточно свободного места – ведь нужно сохранить состояние оперативной памяти. Если в вашем компьютере установлено 8 Гб оперативной памяти, то на жестком диске должно быть как минимум 8 Гб свободного пространства для перехода в режим гибернации.

Откройте главное меню и выберите команду **Панель управления**. Откроется *панель управления системой* (рис. 2.21)

Раздел **Оформление Fly** позволяет изменить оформление рабочего стола – выбрать другие обои, настроить блокировку экрана, выбрать тему оформления, выбрать различные графические эффекты.

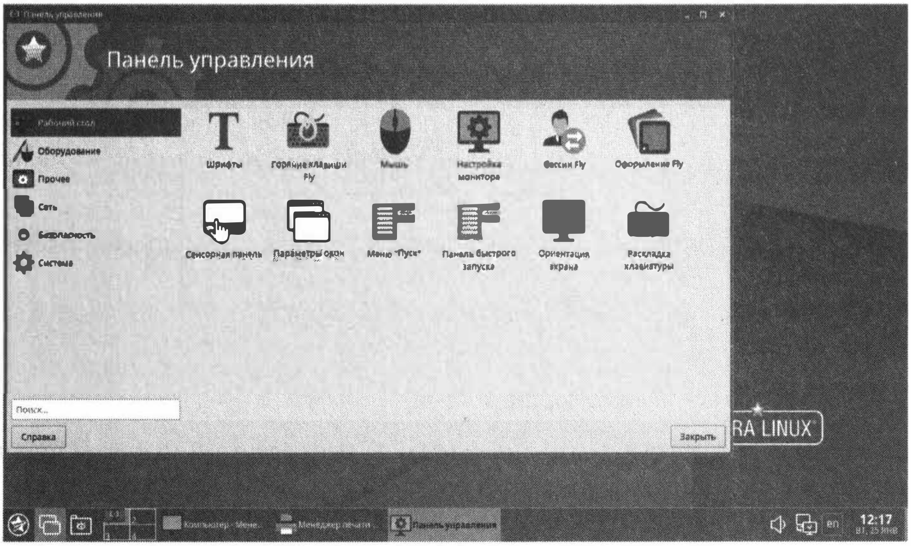


Рис. 2.21. Панель управления

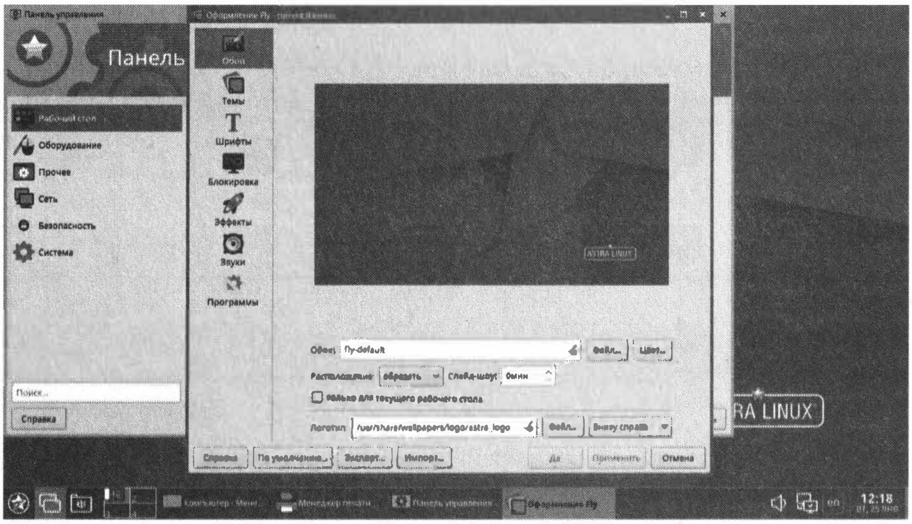


Рис. 2.22. Изменение оформления

Раздел **Настройка монитора** позволяет выбрать разрешение монитора (рис. 2.23).

В разделе **Безопасность** вы найдете утилиту **Политика безопасности**, позволяющую управлять учетными записями пользователями. Утилита **Изменить пароль** позволяет сменить пароль текущего пользователя.

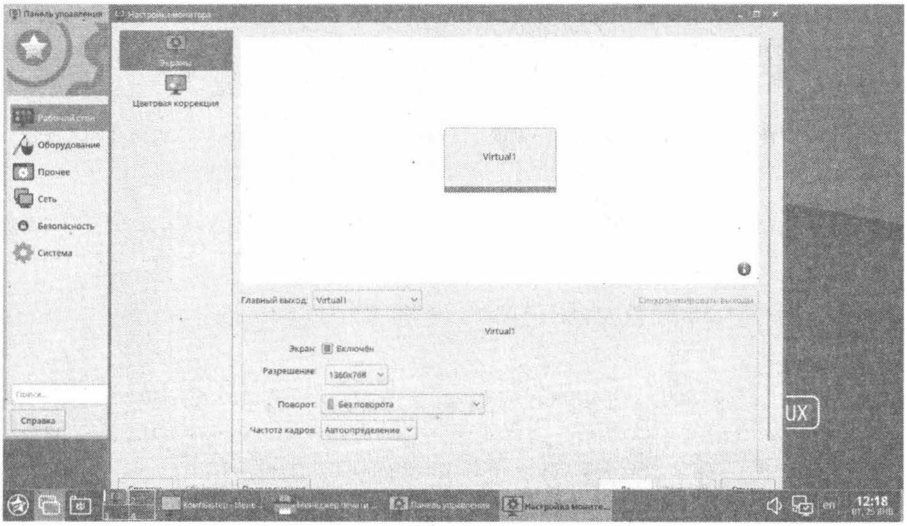


Рис. 2.23. Выбор разрешения монитора

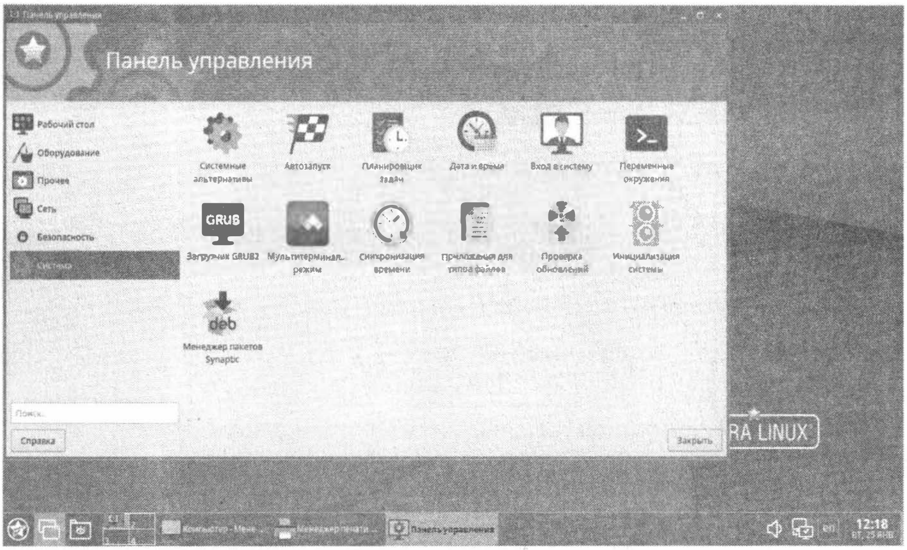


Рис. 2.24. Раздел **Система**

В разделе **Система** много чего интересного. Здесь вы найдете следующие утилиты:

- **Автозапуск** – позволяет управлять автозапуском различных приложений GNOME;

- **Планировщик заданий** – позволяет просматривать и управлять задачами *cron*, что очень удобно и подобных решений нет в других дистрибутивах;
- **Загрузчик GRUB2** – управляет параметрами загрузчика GRUB2 без необходимости редактирования его файла конфигурации вручную;
- **Система инициализации** – позволяет управлять системными службами.

2.3. Автоматический вход в систему

Если вы не страдаете паранойей, существует возможность настройки автоматического входа в систему. При этом система не будет запрашивать пароль, а будет обеспечивать вход выбранного пользователя в систему сразу после загрузки. Для пользователя домашнего компьютера, который работает в гордом одиночестве – это идеальный вариант. Конечно, если этому пользователю нечего скрывать от других членов семьи:)

Для настройки автоматического входа в Astra Linux выполните следующие действия:

1. Откройте **Панель управления;**
2. Перейдите в раздел **Система;**
3. Вызовите утилиту **Вход в систему**;
4. Перейдите на вкладку **Дополнительно**, включите параметр **Разрешить автоматический вход в систему**;
5. Выберите пользователя, автоматический вход в систему которого нужно обеспечить;
6. Нажмите кнопку **Да**;
7. Перезагрузите систему.

В Ubuntu нужно выполнить следующие операции:

1. Откройте экран **Настройки**;
2. Перейдите в раздел **Пользователи**;
3. Нажмите кнопку **Разблокировать** для разблокирования интерфейса управления учетными записями;

4. Выберите пользователя, автоматический вход которого нужно обеспечить;
5. Включите параметр **Автоматический вход**
6. Закройте окно **Настройки**;
7. Перезагрузите систему.

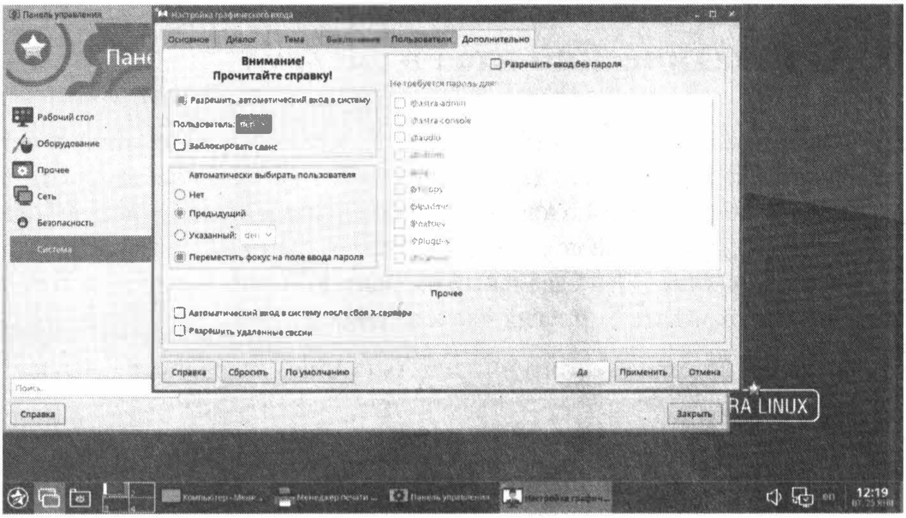


Рис. 2.25. Настройка автоматического входа в Astra Linux

Примечание. Дистрибутив Astra Linux очень понравился и это не реклама. Не смотря на неказистый интерфейс (современные версии Ubuntu выглядят довольно презентабельно, а Astra Linux чем-то похож на дистрибутивы начала 00-вых), он довольно удобен в настройке, а подобной панели управления очень не хватает в той же Ubntu. Продумано множество мелочей, тот же консольный вход при входе в систему – видно, что дистрибутив создавался Linux-пользователем для Linux-пользователя.

2.4. Завершение работы из консоли

Ранее было рассказано, как завершить работу в графическом режиме. Но Linux поддерживает ряд команд, позволяющих завершить работу системы из консоли. Данные команды вы можете, как вводить вручную, так и использовать их в сценариях командной оболочки.

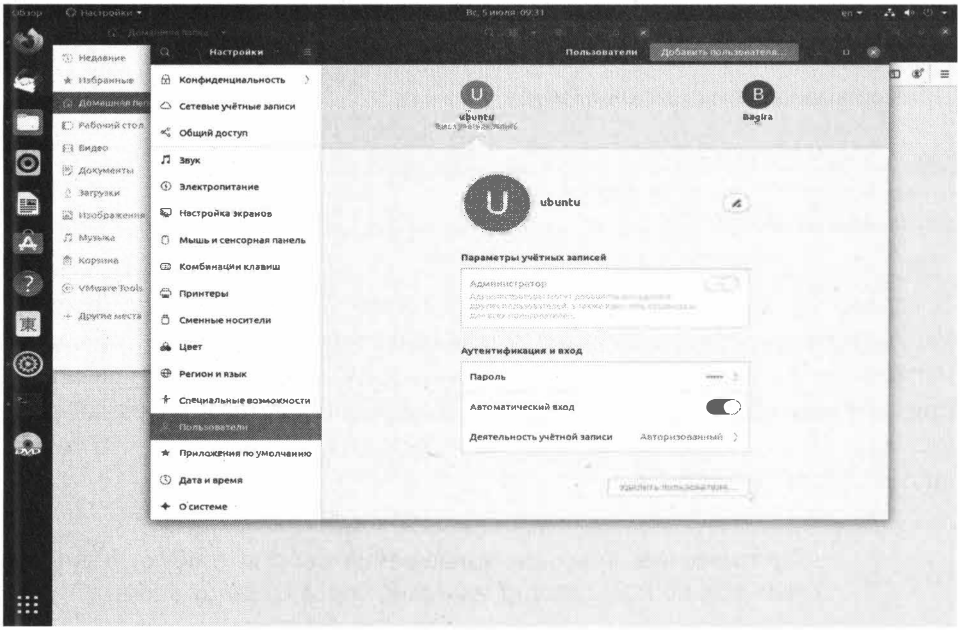


Рис. 2.26. Настройка автоматического входа в Ubuntu

Для завершения работы в Linux используются следующие команды:

- `poweroff` – завершает работу системы Linux и отключает питание компьютера;
- `halt` – завершает работу системы, но не отключает питание;
- `reboot` – перезагружает систему;
- `shutdown` – обеспечивает более гибкое завершение работы.

При завершении работы (в том числе и при перезагрузке) производится ряд очень важных действий, а именно синхронизация буферов ввода/вывода и размонтирование смонтированных файловых систем. Именно поэтому очень важно правильно завершить работу системы.

Программа `shutdown` может не просто завершить работу системы, а сделать это в указанное время. Причем всем зарегистрированным в системе пользователям на их консоль будет отправлено сообщение (определяется администратором) о необходимости завершения работы или перезагрузки. Формат вызова команды `shutdown` следующий:

```
shutdown [параметры] [время] [сообщение]
```

Пример вызова команды *shutdown*:

```
sudo shutdown -r now Bye!
sudo shutdown -h 19:00
```

В первом случае система будет перезагружена (-r) моментально (время – *now*), а всем пользователям будет отправлено сообщение "Bye!". Сообщение не обязательно и вы можете его не указывать, что и продемонстрировано на примере второй команды. Во втором случае работа системы будет завершена (-h) в 19:00. Пользователи получат стандартное сообщение, что работа системы будет завершена.

Примечание. Команды завершения работы требуют полномочий *root*, поэтому вводит их нужно через команду *sudo*.

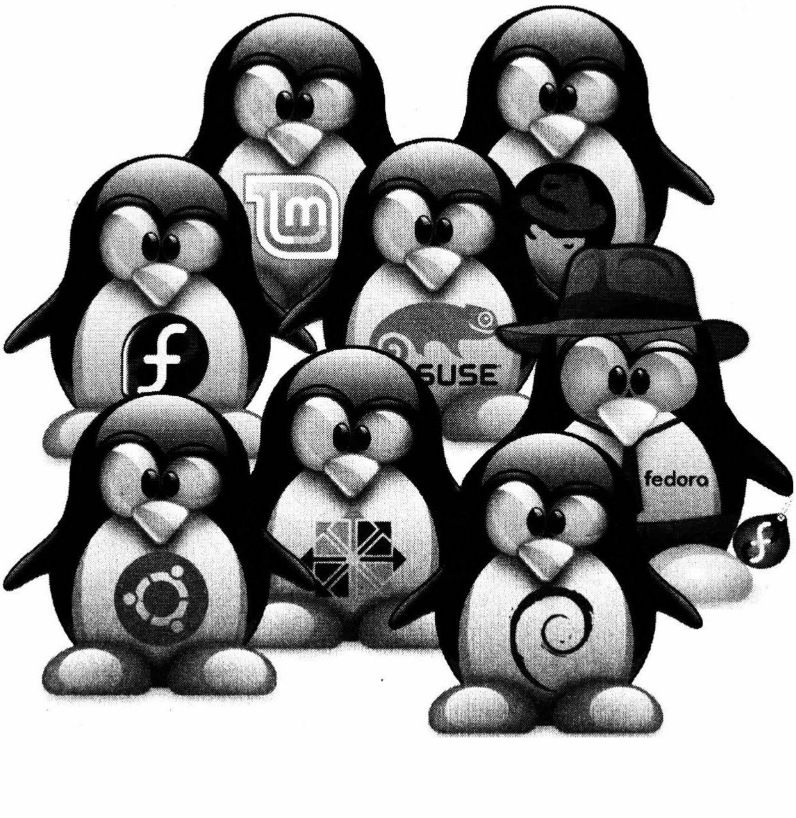
Дополнительные параметры *shutdown* приведены в таблице 2.1.

Таблица 2.1. Параметры команды shutdown

Параметр	Описание
-H (или --halt)	Завершает работу системы, питание не отключается
-P (или --poweroff)	Завершает работу системы, питание отключается
-r (или --reboot)	Перезагружает компьютер
-h	Аналогично --poweroff, то есть завершение работы с отключением питания
-k	Работа системы не завершается, просто каждый пользователь получит указанное сообщение
--no-wall	При завершении работы системы (в том числе перезагрузке) пользователям не будут выводиться
-c	Отменяет отложенное завершение работы (если вы передумали завершать работу или перезагружать систему, но при условии, что процесс завершения работы еще не начат)

Глава 3.

Сразу после установки



Есть некоторые вещи, которые нужно настроить сразу после установки системы. В прошлой главе мы разобрались, как войти в систему, как завершить ее работу, как вызвать средства конфигурации системы. В этой главе мы изменим под себя некоторые параметры системы и установим некоторое важное программное обеспечение. Подробно об установке программ мы поговорим в следующих главах.

3.1. Проверяем и устанавливаем обновления

Очень важно поддерживать систему в актуальном состоянии. Если вы не выбрали установку обновлений при установке системы или не вы устанавливали систему, самое время проверить наличие обновлений.

В Astra Linux щелкните по значку средства обновлений, откроется окно со списком пакетов, требующих обновления. Нажмите **Да**, чтобы произвести обновление системы.

В Ubuntu нажмите Alt + F2, в появившемся окне введите команду *update-manager*. Откроется окно менеджера обновлений (рис. 3.2). Менеджер произведен поиск обновлений и, если таковые будут найдены, предложит их установить. Обычно обновления в Ubuntu устанавливаются автоматически, поэтому при вызове менеджера вы можете часто увидеть картину, изображенную на рис. 3.2. Она означает, что обновления уже установлены и для их применения нужно перезагрузить компьютер.

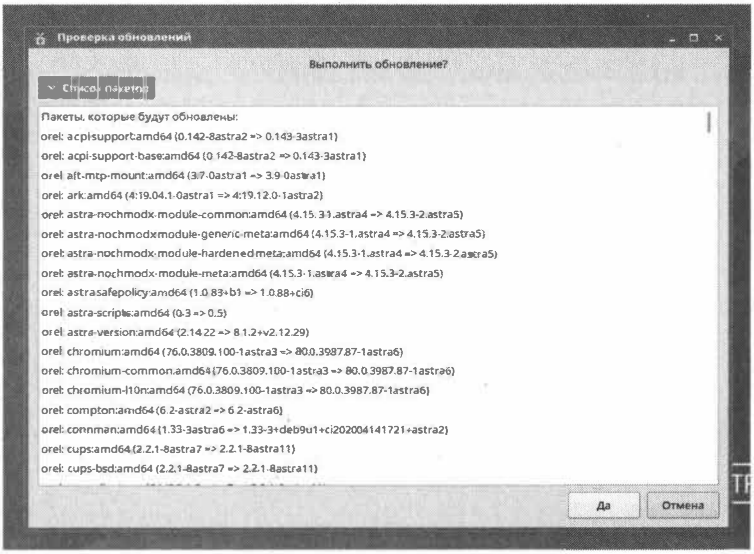


Рис. 3.1. Обновление системы в Astra Linux

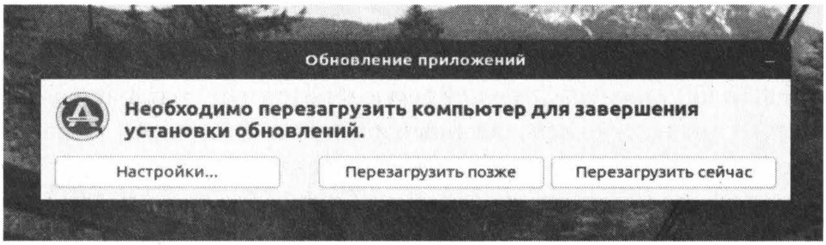


Рис. 3.2. Менеджер обновлений в Ubuntu

Нажмите кнопку **Настройки** (рис. 3.3), чтобы настроить периодичность проверки обновлений. Если вы не хотите, чтобы система автоматически проверяла наличие обновлений, из списка **Автоматически проверять наличие обновлений** выберите **Никогда**.

Если вы предпочитаете командную строку, то откройте терминал и введите команду (для полного обновления системы):

```
sudo apt update && sudo apt full-upgrade
```

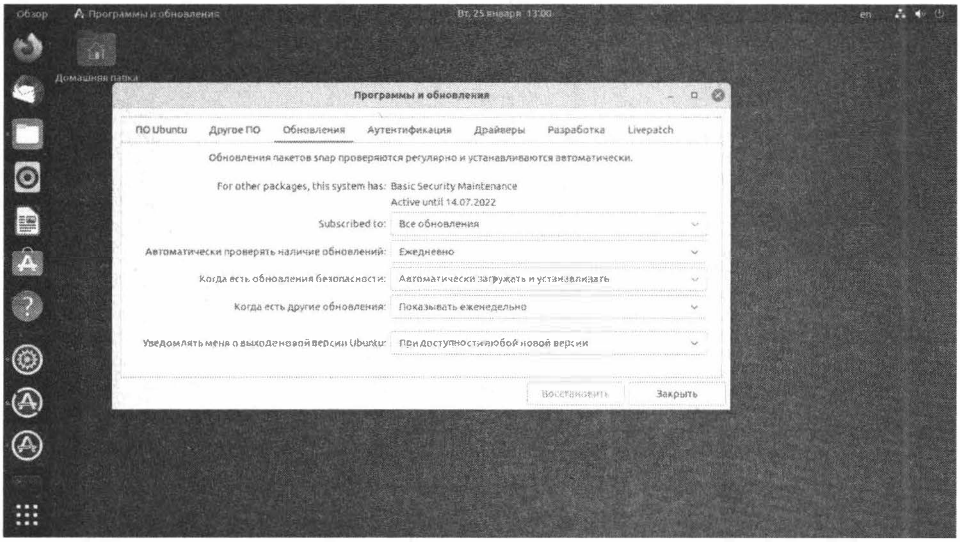


Рис. 3.3. Параметры обновлений по умолчанию

3.2. Настройке Livepatch (только для Ubuntu)

Livepatch (или *Canonical Livepatch Service*) позволяет пользователям Ubuntu применять критические исправления ядра без перезагрузки. Livepatch также помогает поддерживать безопасность вашей системы, применяя обновления безопасности без перезагрузки системы. Сервис бесплатный (до 3 компьютеров) и все, что вам нужно для его активации – настроить учетную запись Ubuntu.

Откройте окно **Программы и обновления** (команда *update-manager*, как было показано ранее) и перейдите на вкладку Livepatch. Нажмите **Войти** для входа в учетную запись Ubuntu или ее создания, а после того, как вход будет выполнен, включите Livepatch, включив единственный переключатель на этой странице. К сожалению, Livepatch доступен не во всех выпусках. В версии 21.10 эта "фича" недоступна. Возможно, она станет доступной для следующего выпуска.

3.3. Отключаем уведомления об ошибках

Для отключения оповещения об ошибках, откройте экран **Настройки**, перейдите в раздел **Конфиденциальность**, затем – в раздел **Диагностика**, для параметра **Отправлять отчеты об ошибках в Canonical** выберите значение **Никогда**.

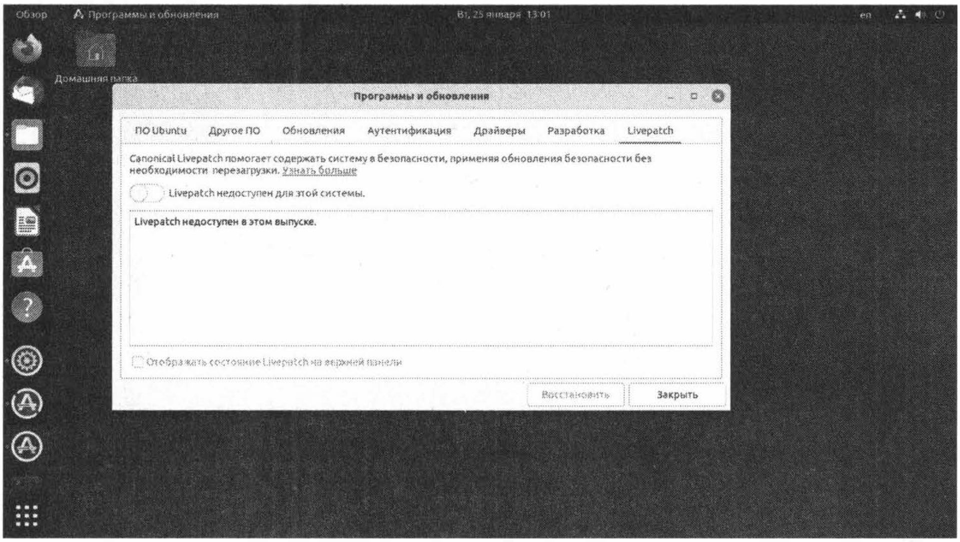


Рис. 3.4. Активация Livepatch

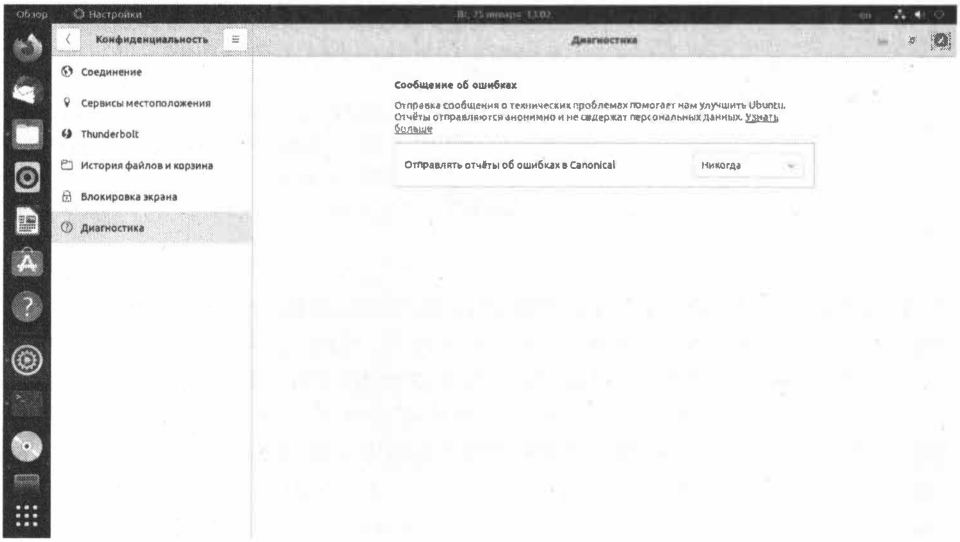


Рис. 3.5. Отключаем уведомления об ошибках

3.4. Настраиваем почтовый клиент

На панели задач Ubuntu есть кнопка вызова почтового клиента Thunderbird. В Astra Linux команда вызова почтового клиента находится в программной группе **Сеть**. Запустите почтовый клиент и просто введите ваш e-mail и

пароль от почтового ящика. Почтовый клиент Thunderbird умный и сам установит остальные параметры почтового сервиса (SMTP, IMAP, порты и т.д.).

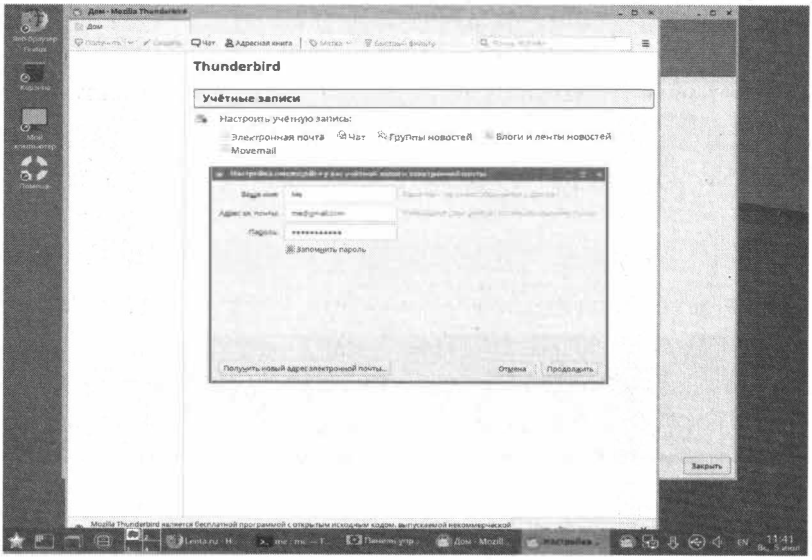


Рис. 3.6. Почтовый агент Thunderbird в Astra Linux

> **Примечание.** Если у вас почтовый ящик на Gmail, нужно в настройках аккаунта включить использование небезопасных приложений – таковыми считаются все не Google приложения, иначе вы не сможете подключить к своему почтовому ящику!

Некоторые пользователи предпочитают использовать веб-интерфейс в браузере, а не почтовую программу. В этом случае можно смело удалить кнопку вызова почтового клиента, чтобы она даром не занимала пространство на панели задач. Для этого щелкните по ней правой кнопкой мыши и выберите команду **Удалить из избранного**. Вы освободите пространство для одной полезной для вас кнопки!

3.5. Установите ваш любимый браузер

По умолчанию в Linux используется браузер Firefox. Это очень хороший браузер, но он нравится далеко не всем. Установить Chrome можно путем загрузки его пакета с официального сайта и его установки. Рассмотрим весь процесс подробнее.

Откройте Firefox и перейдите по ссылке *https://www.google.com/intl/ru/chrome*

1. Нажмите кнопку **Скачать Chrome** (рис. 3.7)
2. Выберите DEB-пакет и нажмите кнопку **Принять условия и установить**.
3. В появившемся окне (рис. 3.8) выберите **Сохранить файл**
4. В окне браузера, когда файл будет скачан, выберите команду **Показать все загрузки** (рис. 3.9)
5. В окне менеджера загрузок нажмите значок папки напротив загруженного DEB-файла, чтобы перейти в папку **Домашняя папка/Загрузки** или откройте файловый менеджер и перейдите в эту папку самостоятельно.
6. Щелкните правой кнопкой мыши и выберите команду **Открыть в терминале** (рис. 3.10)
7. Введите команду *sudo dpkg –i *.deb* (убедитесь, что в каталоге **Загрузки** у вас нет других deb-файлов, поскольку эта команда устанавливает все deb-файлы из текущей папки, что может быть нежелательно)
8. Введите пароль своего пользователя и дождитесь завершения установки. Пакет довольно большой (65 Мб), поэтому его установка может занять несколько минут.
9. Откройте экран **Приложения** (кнопка в самом низу на панели задач) и щелкните правой кнопкой на Chrome (рис. 3.12).
10. Выберите команду **Добавить в избранное**, чтобы добавить кнопку запуска нового браузера на панель задач.
11. Осталось дело за малым – использовать Chrome (рис. 3.13)

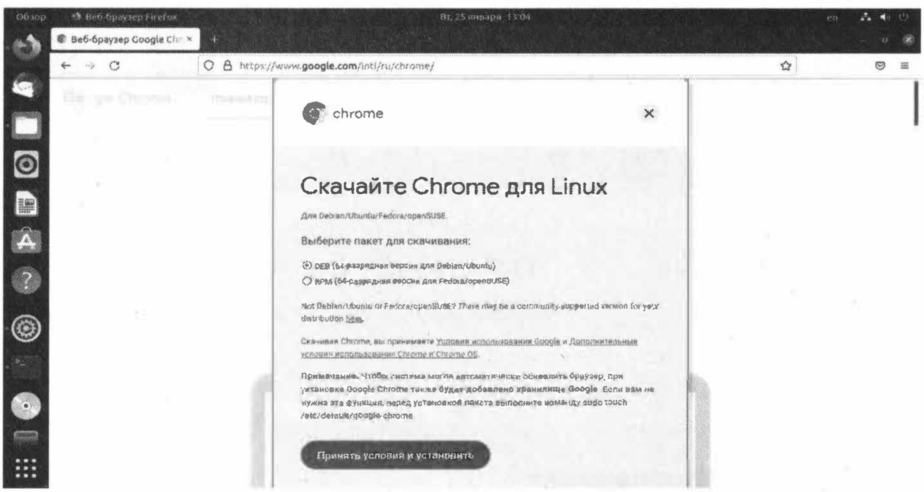


Рис. 3.7. Страница загрузки Chrome

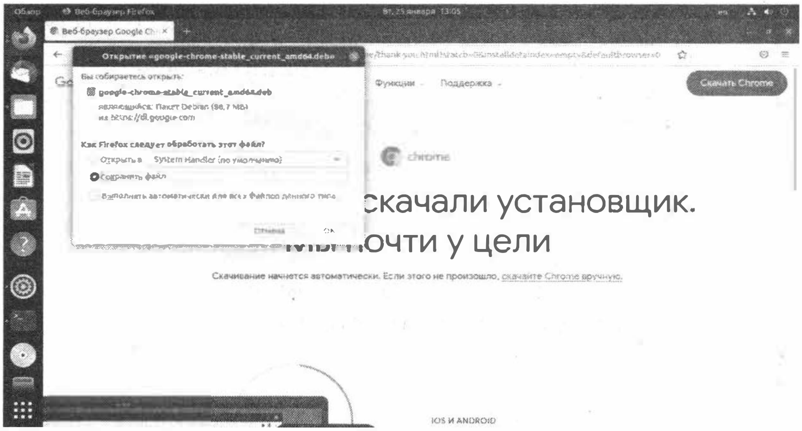


Рис. 3.8. Сохранение файла

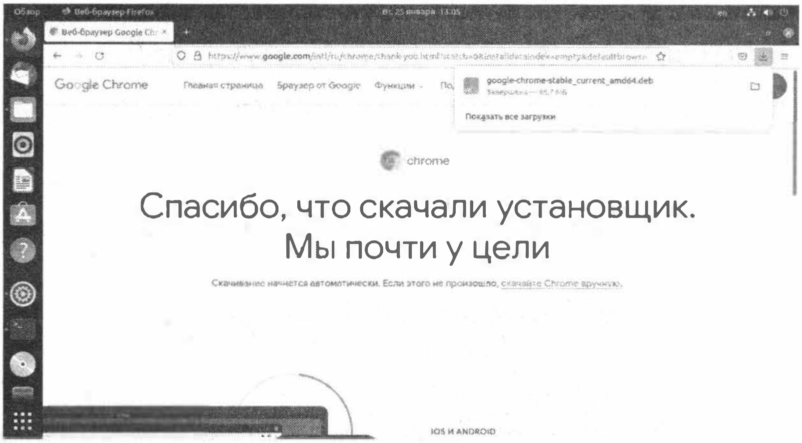


Рис. 3.9. Загрузка завершена

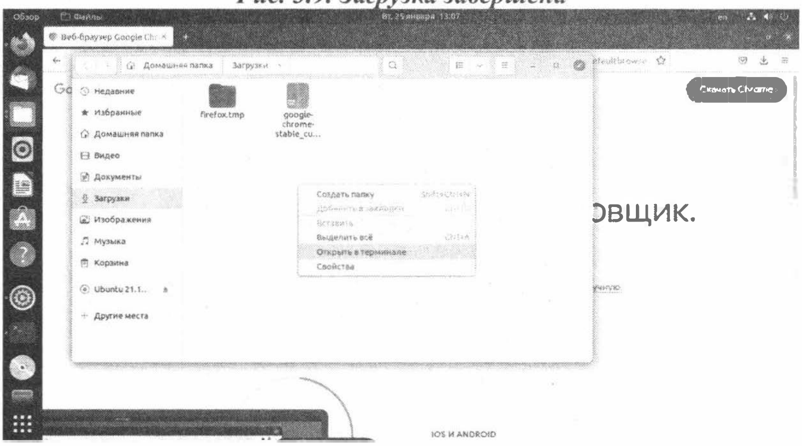


Рис. 3.10. Папка Загрузки



Рис. 3.11. Chrome установлен

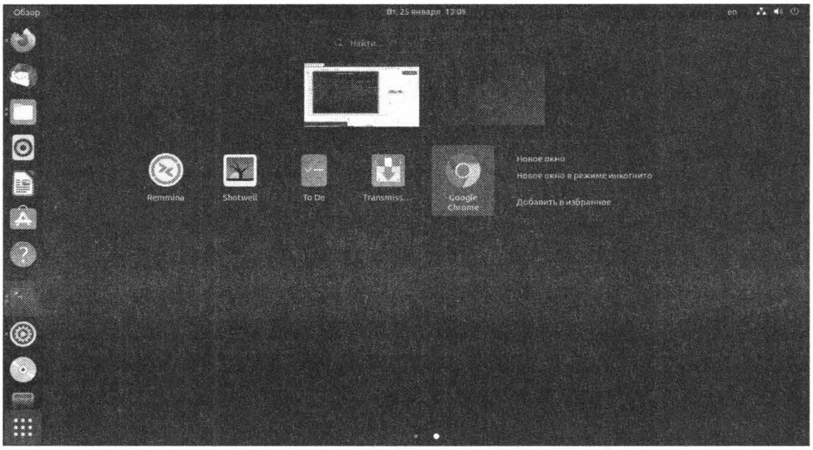


Рис. 3.12. Запуск Chrome

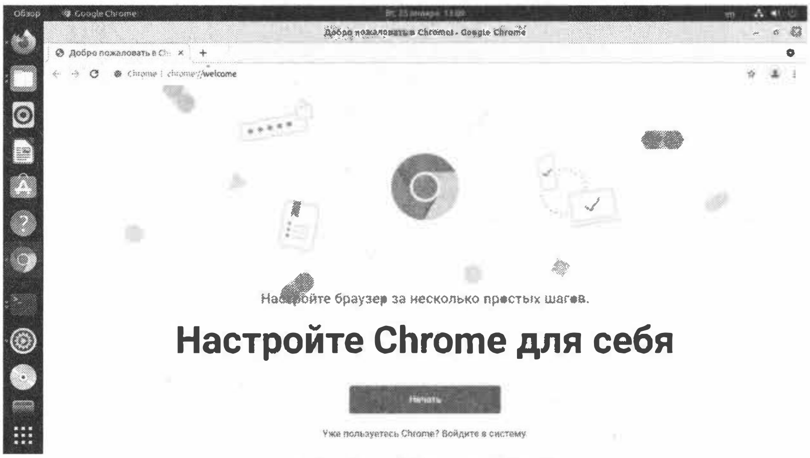


Рис. 3.13. Chrome в Linux

3.6. Установка проигрывателя VLC

VLC – культовый медиа-проигрыватель, поддерживающий множество самых разных медиа-форматов. В Ubuntu для его установки нужно ввести команду:

```
$ sudo snap install vlc
```

Теперь проигрыватель распространяется в виде снапа, а не пакета, что упрощает его установку.

Пользователям Astra Linux повезло больше: VLC уже установлен по умолчанию, поэтому все, что нужно – запустить его из программной группы **Мультимедиа**.



Рис. 3.14. Проигрыватель VLC

3.7. Установка кодеков

Разработчики Ubuntu включают в состав дистрибутива только бесплатное ПО с открытым исходным кодом. К таковому не относятся кодеки – специальное ПО для кодирования/декодирования мультимедиа-форматов. Кодеки бесплатные, но их разработчики не хотят публиковать исходные коды, поэтому по умолчанию кодеки не включены в состав Ubuntu. Для воспроизведе-

ния распространенных аудио- и видеофайлов, таких как MP3, AVI, MPEG4 нужно установить кодеки вручную.

Чтобы установить их, вам нужно установить метапакет **ubuntu-limited-extras**, выполнив следующую команду:

```
$ sudo apt install ubuntu-restricted-extras
```

3.8. Включение ночного режима

Чтобы глаза меньше уставали при работе ночью, рекомендуется включить ночную подсветку, которая сделает цвета более теплыми. Для этого откройте экран **Настройки**, перейдите в раздел **Настройка экранов**, перейдите на вкладку **Ночная подсветка** и активируйте единственный доступный переключатель. Остальные параметры можете оставить без изменений (рис. 3.15).

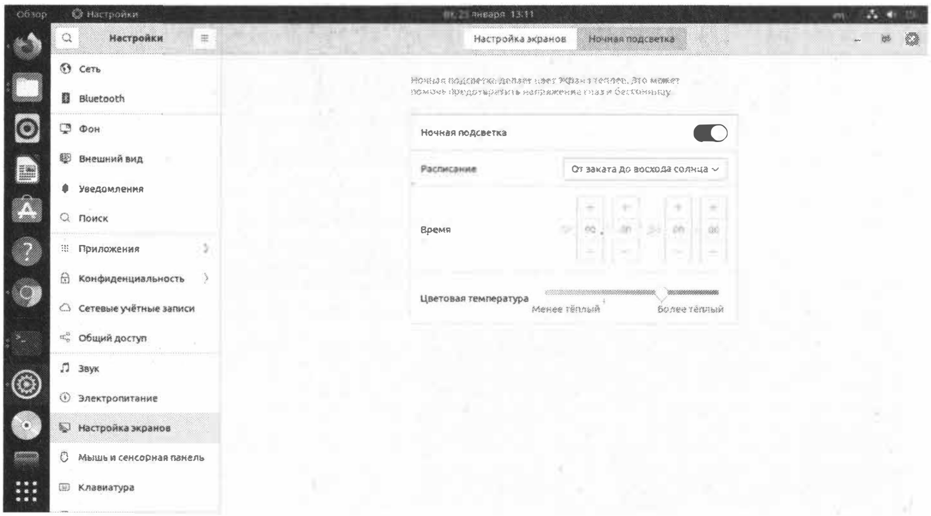


Рис. 3.15. Активация ночного режима

В Astra Linux, к сожалению, подобного режима нет. В нем есть возможность цветовой коррекции, и вы сами можете установить более теплые цвета, но автоматической смены оттенков там нет, и вам придется менять цветовые настройки дважды в сутки – днем и ночью, что очень неудобно.

3.9. Установка wine для запуска Windows-приложений

Подробно о запуске Windows-приложений в Linux мы поговорим далее, а пока установите Wine – средство, обеспечивающее запуск Windows-приложений. Для его установки введите команду:

```
$ sudo apt install wine winetricks
```

3.10. Установка дополнительных архиваторов

Поддержка дополнительных форматов архивов никогда не бывает лишней. Для установки дополнительных архиваторов введите команду:

```
$ sudo apt install rar unrar p7zip-full p7zip-rar
```

3.11. Попробуйте другие графические окружения

Ubuntu поставляется только с рабочим столом Gnome, но вы можете установить другие графические окружения и выбрать то, которое больше нравится вам. Например, следующая команда устанавливает графическую среду Cinnanmon:

```
$ sudo apt install cinnamon-desktop-environment
```

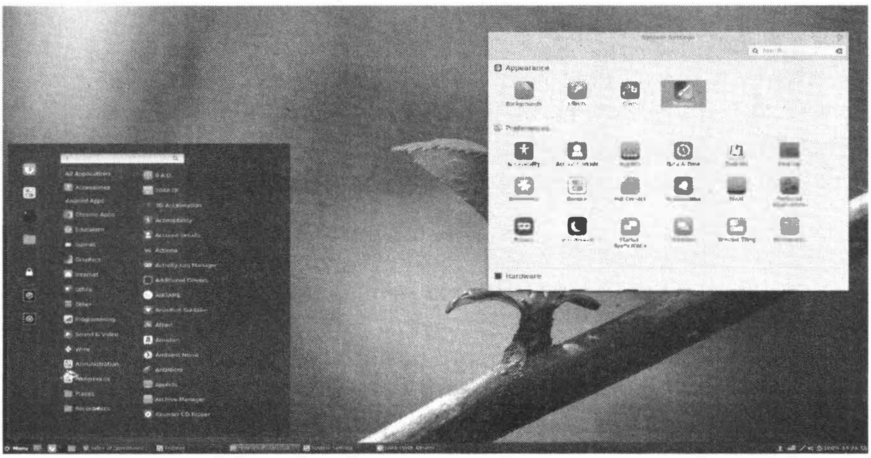


Рис. 3.16. Графическая среда Cinnanmon

А эти команды устанавливают графическую среду MATE:

```
$ sudo apt install tasksel
$ sudo tasksel install ubuntu-mate-desktop
```

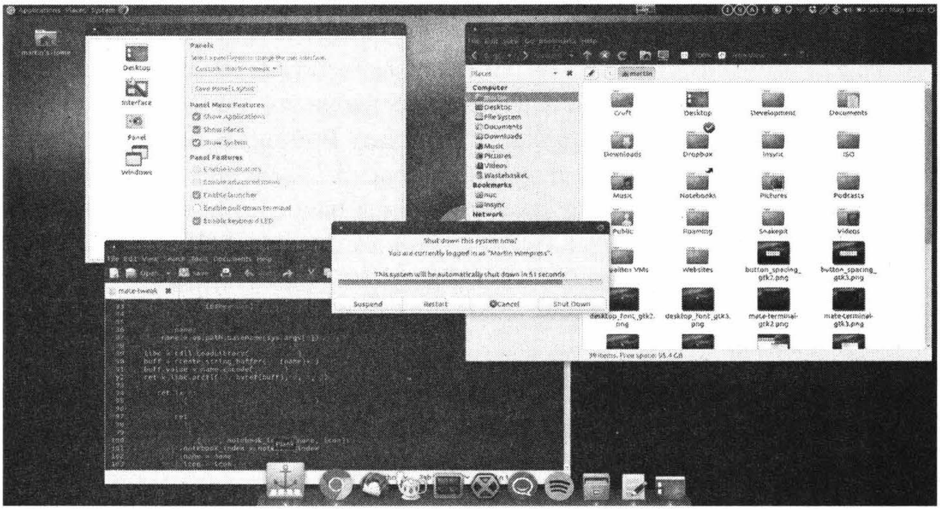


Рис. 3.17. Графическая среда MATE

Какую из них использовать, дело вкуса и здесь каждый решает сам.

3.12. Тонкая настройка GNOME. Установка темы оформления в стиле macOS

Множество настроек графической среды GNOME скрыто от глаз пользователя. Для более тонкой настройки GNOME вы можете использовать утилиту Gnome Tweaks, позволяющую легко кастомизировать ваш **Рабочий стол**. Для ее установки введите команды:

```
$ sudo add-apt-repository universe
$ sudo apt install gnome-tweak-tool
```

Первая команда включает репозитарий *universe*, в котором находится нужный нам пакет. Вполне возможно, что он уже включен у вас, но лучше убедиться в этом явно. Вторая – устанавливает сам пакет.

Далее запустите средство командой или выберите из меню команду **Дополнительные настройки GNOME**:

```
$ gnome-tweaks
```

Данная утилита – настоящая находка для любителей кастомизации (рис. 3.18). Кстати, только с ее помощью можно изменить тему оформления в Ubuntu на любую другую. Перейдите в раздел **Внешний вид** и выберите другую тему оформления, как показано на рис. 3.19. На рис. 3.20 показано, что тема изменена. Далее будет показан пример наиболее популярного "тюнинга" Ubuntu – установка темы оформления в стиле macOS.



Рис. 3.18. Приложение Gnome Tweaks

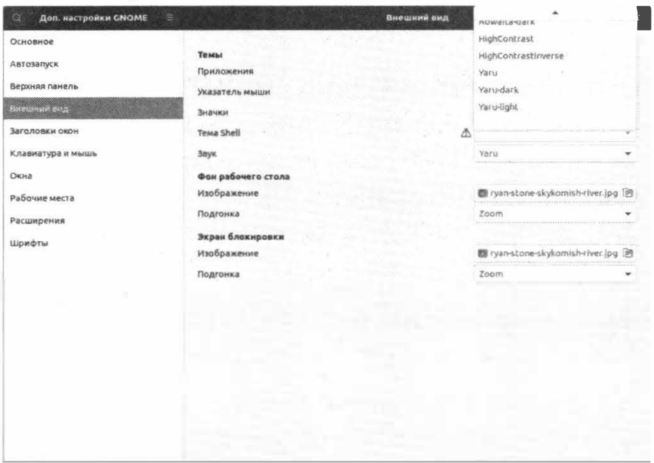


Рис. 3.19. Изменение темы оформления

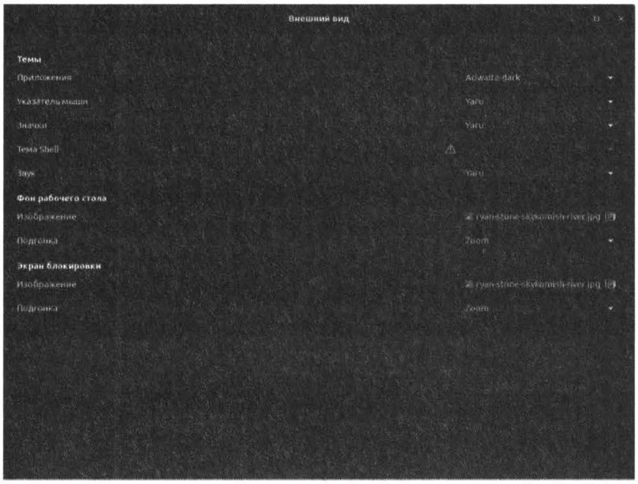


Рис. 3.20. Тема оформления изменена

Установить новую тему достаточно просто. Скачайте архив с темой. Много тем оформления доступно на сайте https://www.gnome-look.org/, например, по адресу https://www.gnome-look.org/p/1275087/ доступна тема в стиле macOS. Перед установкой этой темы нужно установить два пакета:

```
$ sudo apt install gtk2-engines-murrine gtk2-engines-pixbuf
```

Далее нужно скачать архив с темой и распаковать его в каталог .themes:

```
$ tar xf Mojave-dark.tar.xz
$ mkdir ~/.themes
$ mv Mojave-dark ~/.themes/
```

Затем откройте Gnome Tweaks и в качестве темы приложений выберите Mojave-dark. Закройте Gnome Tweaks.

Следующий шаг – скачать значки в стиле macOS. Они доступны по адресу https://www.gnome-look.org/p/1210856/. Аналогично, значки нужно распаковать:

```
$ tar xf Mojave-CT-Night-Mode.tar.xz
```

```
$ mkdir ~/.icons
$ mv Mojave-CT-Night-Mode ~/.icons/
```

После этого опять запустите Gnome Tweaks и в качестве темы значков выберите Mojave-CT-Night-Mode. Наконец, нужно установить тему для курсоров мыши. Скачайте архив по адресу https://www.gnome-look.org/p/1148748/ и распакуйте архив в соответствующий каталог:

```
$ unzip -qq macOS\ Cursor\ Set.zip
$ mv macOS\ Cursor\ Set ~/.icons/
```

Опять запустите Gnome Tweaks и выберите MacOS Cursor Set в качестве темы курсоров.

На рис. 3.21 показан процесс распаковки необходимых архивов, а на рис. 3.22 – настройки, сделанные в Gnome Tweaks. У вас должно получиться так, как показано на рис. 3.22. Обратите внимание, как изменились значки в заголовках окон и значки приложений на панели задач.

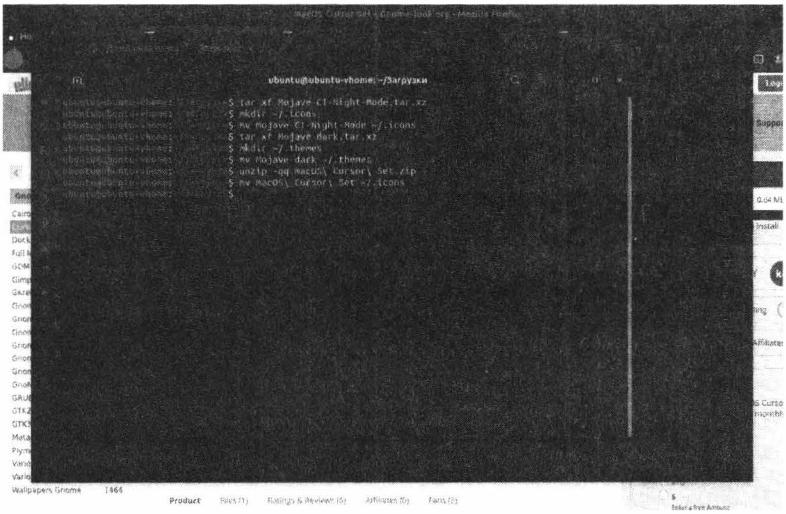


Рис. 3.21. Распаковка ресурсов

Следующий шаг (по желанию) – скачать и установить в качестве фонового следующее изображение: https://www.reddit.com/r/wallpapers/comments/e4fz6s/a_more_purpleish_version_of_the_mac_os_mojave/

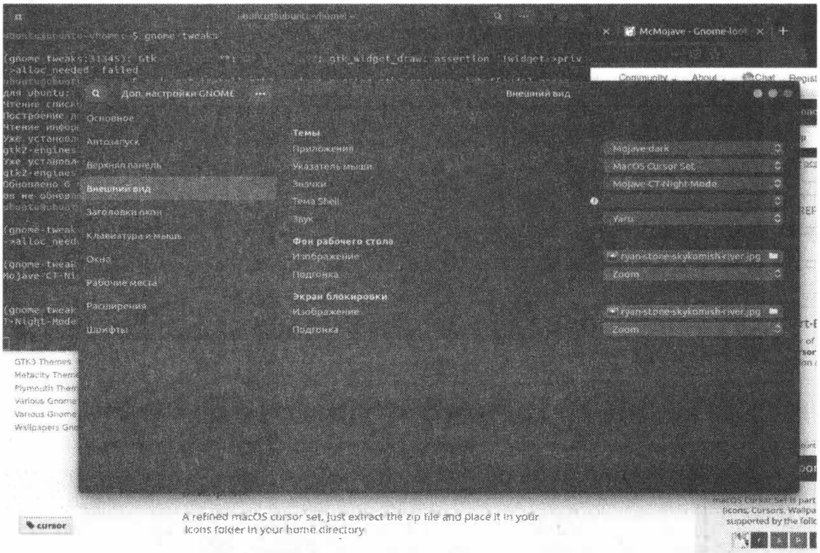


Рис. 3.22. Настройки в Gnome Tweaks

Щелкните правой кнопкой по файлу изображения в папке **Загрузки** и выберите команду **Установить как фон**.



Рис. 3.23. Как будет выглядеть ваша Ubuntu после установки фонового изображения

Вишенка на торте – панель задач в стиле macOS. Установить ее можно командой:

```
$ sudo apt install plank
```

Запустите новую панель задач:

```
plank
```

В нижней части экрана вы увидите ту самую панель (рис. 3.24).

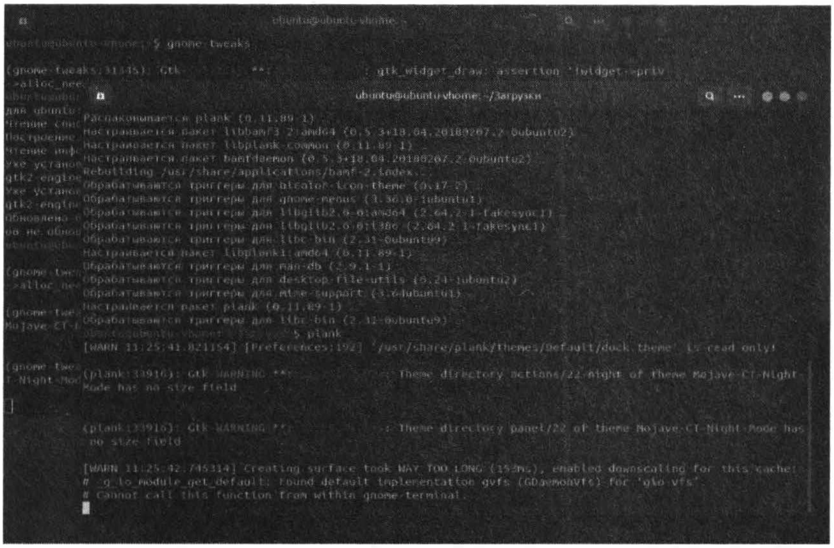


Рис. 3.24. Панель задач в стиле macOS

Откройте Gnome Tweaks и перейдите в раздел **Автозапуск**. Нажмите кнопку + и добавьте в автозапуск приложение Plank. Так мы обеспечим автоматический запуск нашей панели при входе в систему пользователя (рис. 3.25).

Осталось удалить стандартную панель задач. Для этого введите команду:

```
$ sudo apt remove gnome-shell-extension-ubuntu-dock
```

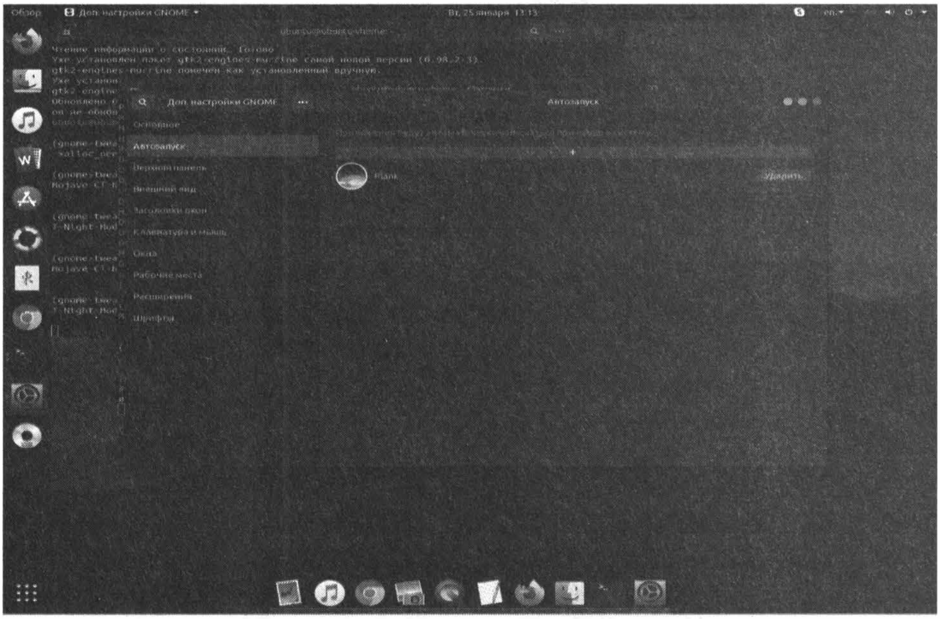


Рис. 3.25. Автоматический запуск панели задач Plank

После этой команды нужно выйти из системы и снова в нее войти. В результате у вас должна получиться "почти" macOS (рис. 3.26).

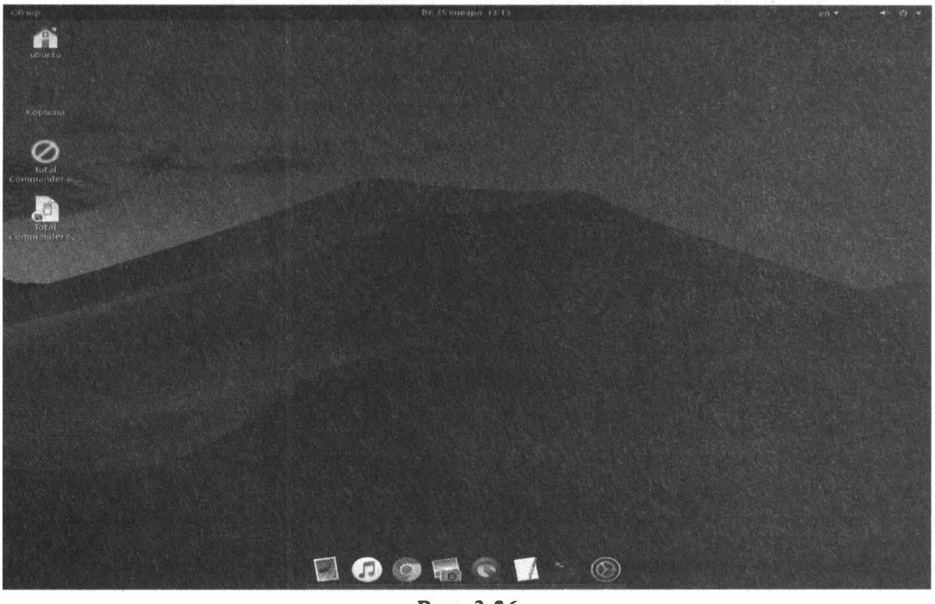
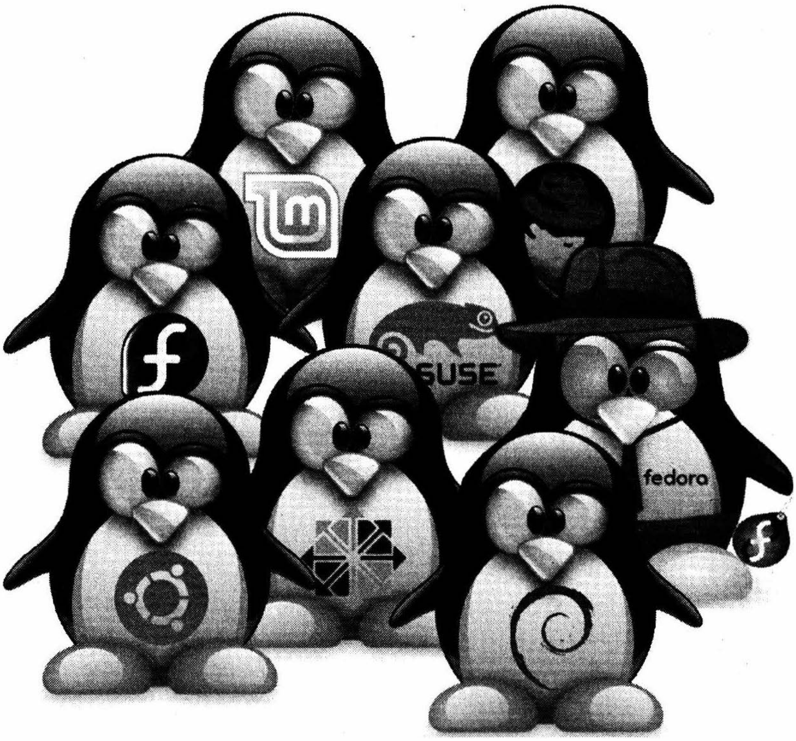


Рис. 3.26

Глава 4.

Основы командной строки



Командная строка – неотъемлемая часть Linux. Посредством командной строки осуществляется выполнение команд Linux. Эффективно сможет работать с Linux только тот, кто освоил принципы работы в командной строке.

4.1. Ввод команд

Ввод команд осуществляется в приложении **Терминал**. Это эмулятор консоли Linux, позволяющий вводить команды. Можно переключиться из графического режима в консоль (нажав Ctrl + Alt + F1) и вводить команды непосредственно в консоли Linux. Но для большинства пользователей будет удобнее работа с эмулятором терминала в графическом режиме. Приложение **Терминал** изображено на рис. 4.1.

Для ввода команды нужно ввести ее в приглашение командной строки и нажать **Enter**. После чего вы увидите результат выполнения команды. Обычно приглашение командной строки имеет вид:

```
пользователь@компьютер:текущий_каталог<$|#>
```

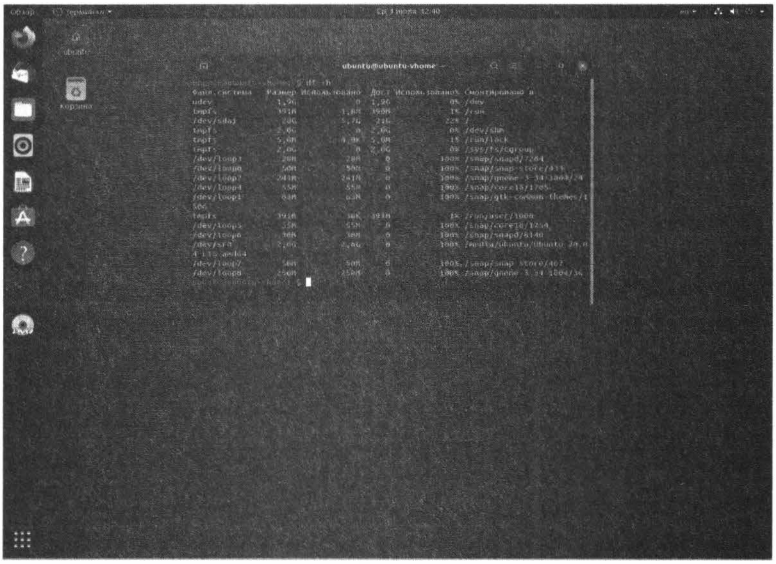


Рис. 4.1. Приложение Терминал (Ubuntu)

Посмотрите на рис. 4.2. В первом случае наш пользователь называется *ubuntu*, имя компьютера *ubuntu-vhome*, каталог ~ (так сокращенно обозначается домашний каталог пользователя), далее следует символ $. Символ $ обозначает, что вводимая команда будет выполняться с привилегиями обычного пользователя.

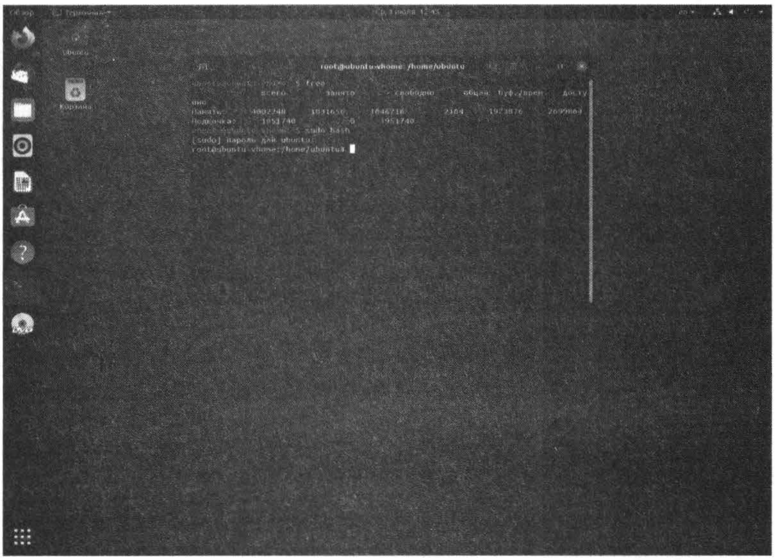


Рис. 4.2. Разные формы приглашения командной строки

Далее мы выполняем команду *sudo bash*. Команда *sudo* запрашивает привилегии суперпользователя *root* для командной оболочки *bash*. По сути, после этого мы получим командную строку с привилегиями *root*. Все вводимые данные команды будут выполняться с максимальными правами.

Посмотрим, как поменялось приглашение командной строки. Мы видим, что пользователь уже не *ubuntu*, а *root*, и что каталог выводится как /home/ubuntu – это домашний каталог пользователя *ubuntu*, в котором мы находимся. Символ ~ не выводится, поскольку домашний каталог пользователя *root* называется /root. Как только мы перейдем в этот каталог, то получим ~ вместо имени каталога. Последний символ # обозначает, что вводимая команда будет выполняться с максимальными правами. С максимальными правами нужно быть осторожнее и стараться, как можно реже использовать данный режим во избежание нанесения системе вреда.

4.2. Автодополнение командной строки

Linux содержит множество команд. Если вы забыли точное название команды или просто хотите ускорить ее ввод, вы можете использовать автодополнение командной строки. Для этого введите первые буквы названия команды и нажмите **Tab**. Далее система или автоматически дополнит команду или выведет список доступных вариантов, если доступно несколько вариантов по введенной команде.

Аналогично, автодополнение может использоваться для имен файлов и каталогов. Например, вы хотите перейти в каталог applications/xnfbdh73/public_html. Вместо того, чтобы вводить этот длинный путь (и помнить его!), вводите команду так:

```
cd a <нажмите Tab> / x <нажмите Tab> / p <нажмите Tab>
```

В конечном итоге вместо ввода 33 символов пути вам нужно будет ввести 5!

4.3. Перенаправление ввода/вывода

Рассмотрим следующую команду:

```
cat very_long_file.txt | less
```

Здесь мы пытаемся просмотреть очень длинный файл very_long_file.txt. Поскольку он очень длинный и не помещается на одном экране, мы перенаправили вывод первой части команды (cat very_long_file.txt) на стандартный вывод команды *less*, которая обеспечивает удобный просмотр длинных файлов.

Синтаксис следующий:

```
команда_1 | команда_2
```

Особых ограничений не существует, и вы можете передать вывод второй команды на ввод третьей и т.д.:

```
команда_1 | команда_2 | команда_3
```

С помощью такого перенаправления можно автоматизировать некоторые команды, что важно в сценариях *bash*, например, вот как можно утвердительно ответить на запрос об удалении файла:

```
echo y | rm file.old
```

Очень часто перенаправление ввода/вывода используется в таком контексте, в котором использовали его мы: большой вывод перенаправляется на программу просмотра (less) или на программу-фильтр, такую как *grep*.

Кроме перенаправления вывода программы на ввод другой программы, его можно перенаправить в файл, например:

```
ps -A > processes.txt
```

Если файл processes.txt не существует, он будет создан. Если существует – перезаписан. Если нужно дописать вывод программы в конец файла, не удаляя существующий файл, тогда используйте два знака больше:

```
ps -A >> processes.txt
```

В этом случае, если файл не существует, то он будет создан, а если существует, то информация будет дописана в конец файла.

4.4. Справочная система *man*

В Linux справочная система всегда под рукой. Например, если вы забыли параметры команды *df*, просто введите команду:

```
man df
```

Откроется страница руководства (в большинстве случаев – на русском языке), в котором будет описаны все возможные параметры по интересующей вас команде и даны рекомендации по их применению. Для работы справочной системы не требуется соединение с Интернетом, поскольку все страницы справочного руководства уже загружены на ваш компьютер.

Далее будут рассмотрены некоторые полезные команды, знание которых просто обязательно для каждого пользователя Linux.

4.5. Команды для работы с файлами и каталогами

4.5.1. Команды для работы с файлами

В каждой операционной системе есть команды для работы с файлами и каталогами. Linux – не исключение. Рассмотрим стандартные команды Linux для работы с файлами (см. табл. 4.1).

Таблица 4.1. Стандартные команды Linux для работы с файлами

Команда	Описание
cat файл	Выводит текстовый файл. Файлы могут быть довольно длинными, поэтому лучше использовать ее в паре с командой *less*, например, cat /etc/services | less

tac файл	Подобна команде cat, но выводит файл в обратном порядке. Данная команда удобна для чтения журналов, в которых самые свежие сообщения заносятся в конец файла, например, tac /var/log/messages | less
cp файл1 файл2	Копирует файл1 в файл2. Если второй файл существует, программа спросит вас, нужно ли его перезаписать
mv файл1 файл2	Используется для перемещения файла1 в файл2. Можно использовать для переименования файлов
rm файл	Удаляет файл
touch файл	Используется для создания нового пустого файла
locate файл	Быстрый поиск файла. Позже мы рассмотрим процесс поиска подробнее
which исполнимый_ файл	Производит быстрый поиск программы (исполнимого файла). Если программа находится в пути PATH, то which выведет каталог, в котором находится программа

В таблице 4.1 представлены основные команды, которые используются для создания (touch), копирования (cp), перемещения (mv) и удаления (rm) файлов, а также несколько дополнительных команд.

Рассмотрим несколько примеров:

```
$ dmesg > kernel.messages
$ cat kernel.messages | less
$ cp kernel.messages krn.msg
$ rm kernel.messages
```

Первая команда выводит загрузочные сообщения ядра в файл kernel.messages. Вторая выводит содержимое этого файла на экран, а команда *less* организует удобный постраничный просмотр этого файла. Далее команда *cp* копирует файл kernel.messages в файл krn.msg, а последняя команда удаляет

наш исходный файл kernel.messages. В принципе, вместо последних двух команд можно было использовать одну команду *mv*:

```
mv kernel.messages krn.msg
```

При указании имени файла вы можете использовать маски * и ?. Как обычно, символ * заменяет любую последовательность символов, а ? – один символ. Например:

```
rm *.tmp
rm /tmp/*
cp *.txt /media/ext-usb
cp ???.txt /media/ext-usb
```

Первая команда удаляет все файлы, заканчивающиеся на ".tmp", в текущем каталоге. Вторая – удаляет все файлы из каталога /tmp. Третья копирует все файлы с "расширением" .txt из текущего каталога в каталог /media/ext-usb. Четвертая команда копирует все файлы, имя которых состоит из трех любых символов и заканчивается на ".txt", например, abc.txt, в каталог /media/ext-usb.

Примечание. Как вы уже догадались, к каталогу /media/ext-usb можно подмонтировать внешний USB-диск и тогда копируемые файлы физически окажутся на внешнем жестком диске. Подробнее о монтировании мы поговорим в других главах.

Примечание. Обратите внимание, что в таблице 4.1 команды представлены без параметров. Хотя дополнительные параметры есть у каждой команды. Вы не обязаны помнить все параметры, для этого есть справочная система ***man***. Вам нужно помнить только названия команд, а параметры вы всегда сможете "подсмотреть" в ***man***.

Мы не рассмотрели команды редактирования текстовых файлов. Они не являются стандартными (кроме программы **vi**, которой пользоваться очень неудобно), но в вашей системе по умолчанию могут быть установлены следующие текстовые редакторы:

- **nano** – удобный текстовый редактор;
- **joe** – небольшой и удобный текстовый редактор;
- **pico** – текстовый редактор, устанавливаемый вместе с почтовым клиентом pine;
- **mcedit** – текстовый редактор, устанавливаемый вместе с файловым менеджером **mc**.

Если у вас нет этих редакторов, вы можете установить их. Например, установите **mc** – и вы получите и файловый менеджер и текстовый редактор сразу:

```
sudo apt install mc
```

4.5.2. Команды для работы с каталогами

Аналогично командам для работы с файлами, команды для работы с каталогами представлены в таблице 4.2.

Таблица 4.2. Стандартные команды Linux для работы с каталогами

Команда	Описание
cd каталог	Изменение каталога
ls каталог	Выводит содержимое каталога
rmdir каталог	Удаляет пустой каталог
rm -r каталог	Рекурсивное удаление непустого каталога
mkdir каталог	Создает каталог
cp каталог1 каталог2	Команду *cp* можно использовать и для копирования каталогов. В данном случае *cp* копирует каталог 1 в каталог 2
mv каталог1 каталог2	Команду *mv* можно использовать и для перемещения каталогов. В данном случае *mv* перемещает каталог 1 в каталог 2

Обратите внимание, что команда *rmdir* не может удалить непустой каталог, поэтому если не хотите удалять сначала файлы и подкаталоги из удаляемого каталога, то лучше использовать команду `rm -r каталог`. Например:

```
$ mkdir /home/bagira/test
$ touch /home/bagira/test/test-file
$ rm -r /home/bagira/test
```

Как и в случае с командой *rm*, вы можете задать параметр *-r* для команд *cp* и *mv*. В этом случае операция копирования или перемещения будет выполняться рекурсивно.

Очень важной операцией является просмотр содержимого каталога, для которой используется команда *ls*. Поэтому сейчас сделаем исключением для этой команды и рассмотрим ее параметры (табл. 4.3). А общий формат вызова этой команды таков:

```
ls [параметры] [каталог]
```

Таблица 4.3. Параметры команды ls

Параметр	Описание
-C	Выводит список файлов в колонках с вертикальной сортировкой
-F	Для каждого каталога добавлять суффикс '/', а для каждого исполняемого файла - '*', а для каждого FIFO-канала - '|'
-R	Рекурсивный вывод, то есть команда *ls* будет выводить не только содержимое каталога, но и подкаталогов
-a	Показывать скрытые файлы
-i	Показывать иноды для каждого файла (будет показан серийный номер файла)
-l	"Длинный" формат вывода, в котором отображается тип файла, права доступа, количество ссылок на файл, имя владельца, имя группы, размер файла, метка времени создания файла и имя файла. В колонке типа файла могут быть следующие значения: d (каталог), b (блочное устройство), c (символьное устройство), l (символическая ссылка), p (FIFO-канал), s (сокет)

-r	Сортировка в обратном порядке

В таблице 4.3 приведены не все параметры команды *ls*, а только самые основные.

При задании имени каталога можно использовать следующие специальные имена:

- . – обозначает текущий каталог.
- .. – обозначает родительский каталог.
- ~ – домашний каталог пользователя, например, если вы вошли под пользователем **bagira**, то путь ~/file.txt равноценен /home/bagira/file.txt.

4.6. Команды системного администратора

Существуют команды, которые нужно знать каждому системному администратору. В этой главе рассматривается необходимый минимум таких команд. Нужно отметить, что команд системного администратора гораздо больше в Linux, по сути, в каждой главе мы рассматриваем те или иные команды администратора. В этой главе мы рассмотрим некоторые базовые команды. Возможно, они не пригодятся вам прямо сейчас, но вы еще не раз вернетесь к этой главе в будущем.

4.6.1. Команды для работы с устройствами и драйверами

В таблице 4.4 приведены некоторые команды, которые помогут вам обнаружить аппаратную проблему – проблему с устройством, либо с его драйвером.

Таблица 4.4. Команды, предоставляющие информацию об устройствах

Команда	Описание
`uname -a`	Очень важная команда, сообщающая версию ядра. Очень важно, чтобы устанавливаемые модули были откомпилированы под вашу версию ядра

lsdev	Выводит информацию об устройствах. По умолчанию эта команда не установлена, нужно установить пакет procinfo
lshal	Выводит параметры всех устройств
lspci, lsusb, lshw	Выводят соответственно список PCI-устройств, USB-устройств и список оборудования компьютера
lsmod	Выводит список загруженных модулей ядра
dmidecode	Отображает информацию о BIOS компьютера
cat /proc/cupinfo	Выводит информацию о процессоре
cat /proc/meminfo	Отображает информацию о памяти
cat /proc/mounts	Показывает точки монтирования
cat /proc/net/dev	Выводит сетевые интерфейсы и статистику по ним
cat /proc/version	Похожа на uname, выводит версию ядра
cat /proc/interrupts	Отображает информацию по прерываниям
cat /proc/swaps	Выводит информацию о файлах подкачки

4.6.2. Команды настройки сетевых интерфейсов

Подробно настройка сети будет рассматриваться далее, а пока рассмотрим таблицу 4.5, в которой представлен короткий список команд, которые вам могут пригодиться при настройке сети.

Таблица 4.5. Некоторые команды настройки сети

Команда	Описание
route	Просмотр и изменение таблицы маршрутизации

`dmesg | less`	Просмотр сообщений ядра, которые выводятся ядром при загрузке системы
`iwconfig`	Выводит информацию обо всех беспроводных интерфейсах
`iwlist scan`	Поиск беспроводных сетей
`dhclient wlan0`	Обновляет IP-адрес и другую сетевую информацию беспроводного интерфейса wlan0
`iwevent`	Просмотреть события беспроводной сети
`sudo /etc/init.d/dbus restart`	Перезапуск всех сетевых служб (работает не во всех дистрибутивах)
`sudo systemctl restart <служба>` или `service <служба> restart`	Перезапуск службы. Например, *sudo systemctl restart networking* перезапускает сеть

4.6.3. Программы тестирования и настройки жесткого диска

Команды для тестирования и настройки жесткого диска, подобно ранее приведенным командам, также представлены в виде таблицы – табл. 4.6.

Таблица 4.6. Команды для тестирования и настройки жесткого диска

Команда	Описание
`badblocks -v <имя_устройства>`	Осуществляет проверку жесткого диска на наличие "плохих" блоков. Параметр *-v* включает подробный отчет
`hdparm`	Тестирование производительности и настройка жесткого диска. Например, параметр *-t* может протестировать производительность (`hdparm -t /dev/sda`), а параметр -E установить скорость привода CD/DVD: `hdparm -E 2 /dev/sr0`

hddtemp	Отображает температуру жесткого диска
bonnie	Тестирует производительность жесткого диска
cpuburn	Тестирование процессора (стресс-тест процессора)
screentest	Тестирование и настройка монитора
smartmontools	SMART-мониторинг. Нужно, чтобы ваши жесткие диски поддерживали S.M.A.R.T

4.7. Команды обработки текста

4.7.1. Редактор *sed*

Команда *sed* – мощный потоковый редактор и ему можно было бы посвятить целую главу, но не вижу такой необходимости, поскольку в современных дистрибутивах имеется документация на русском языке. Главное знать, что такая программа есть. А чтобы вы заинтересовались, давайте рассмотрим несколько примеров использования этой программы:

Заменить строку "string1" на ""string2" в файле report.txt, результат вывести на стандартное устройство вывода:

```
sed 's/stringa1/stringa2/g' report.txt
```

вывести пятую строку файла report.txt:

```
sed -n '5p;5q' report.txt
```

Удалить пустые строки из файла:

```
sed '/^$/d' report.txt
```

Удалить строку "string1" из текста, не изменяя всего остального:

```
sed -e 's/string1//g' report.txt
```

Удалить пустые символы в в конце каждой строки:

```
sed -e 's/ *$//' report.txt
```

Удалить пустые строки и комментарии из файла:

```
sed '/ *#/d; /^$/d' report.txt
```

Преобразовать символы из нижнего регистра в верхний:

```
echo 'test' | tr '[:lower:]' '[:upper:]'
```

Удалить первую строку из файла:

```
sed -e '1d' report.txt
```

4.7.2. Подсчет количества слов/символов

Команда *wc* используется:

- для подсчета слов в текстовом файле:

  ```
  wc /var/log/messages
  ```

- для подсчета количества строк (если задан параметр -l):

  ```
  wc -l /var/log/messages
  ```

- для подсчета количества символов (параметр -c):

  ```
  wc -c /var/log/messages
  ```

4.7.3. Сравнение файлов

Команда *cmp* используется для сравнения двух файлов. Если файлы идентичны, то *cmp* вообще никак не реагирует. А вот если файлы отличаются, то *cmp* выводит номер строки и номер символа в строке, откуда начинается различие.

Команда *cmp* более универсальна, поскольку она может использоваться как для сравнения текстовых, так и двоичных файлов. А вот команда *diff* и ее аналоги умеют сравнивать только текстовые файлы.

Формат вызова команды следующий:

```
cmp [параметры] файл1 файл2
```

Параметры команды *cmp* указаны в табл. 4.7.

Таблица 4.7. Параметры команды cmp

Параметр	Описание
-c	Вывод отличающихся символов
-i n	Игнорировать первые n символов
-l	Вывод позиций всех отличий, а не только первого
-s	Не выводить информацию на экран, при этом код возврата будет следующим: 0 — файлы одинаковые; 1 — файлы отличаются; 2 — ошибка при открытии одного из файлов

4.7.4. Разбивка текста на колонки

Команда *column* используется для разбивки текста на несколько столбцов. Текст может быть прочитан как из файла, так и со стандартного ввода, если файл не указан.

Формат вызова команды:

```
column [параметры] [файл]
```

Параметры команды *column* приведены в табл. 4.8.

Таблица 4.8. Параметры команды column

Параметр	**Описание**
-c n	Задает количество столбцов (число *n*)
-s символ	Указанный символ будет использоваться в качестве разделителя столбцов
-t	Текст будет форматироваться как таблицы. По умолчанию разделителем полей считается пробел, но с помощью параметра *-s* можно задать другой разделитель
-x	Сначала будут заполняться столбцы, а потом строки

4.7.5. Команды *diff* и *diff3*

Команда используется для сравнения двух файлов. Формат вызова программы *diff*:

```
diff [параметры] файл1 файл2
```

В выводе программы отличающиеся строки помечаются символами > и <:

- строка из первого файла помечается символом <;
- строка из второго файла — символом >.

Самые полезные параметры программы *diff* приведены в табл. 4.9.

Таблица 4.9. Параметры команды diff

Параметр	Описание
-a	Сравнение всех файлов, в том числе бинарных
-b	Программа будет игнорировать пробельные символы в конце строки
-B	Игнорирует пустые строки
-e	Применяется для создания сценария для редактора *ed*, который будет использоваться для превращения первого файла во второй
-w	Игнорирует пробельные символы
-y	Вывод в два столбца
-r	Используется для сравнения файлов в подкаталогах. Вместо первого файла указывается первый каталог, вместо второго файла — соответственно второй каталог

Команда *diff3* похожа на *diff*, только используется для сравнения трех файлов. Формат вызова команды таков:

```
diff3 [параметры] файл1 файл2 файл3
```

Программа выводит следующую информацию:

- ==== — все три файла разные;
- ===1 — первый файл отличается от второго и третьего;
- ===2 — второй файл отличается от первого и третьего;
- ===3 — третий файл отличается от первого и второго.

Параметры команды *diff3* указаны в таблице 4.10.

Таблица 4.10. Параметры команды diff3

Параметр	Описание
-a	Сравнивать файлы как текстовые, даже если они являются бинарными

-A	Создание сценария для редактора ed, который показывает в квадратных скобках все отличия между файлами
-e	Создает сценарий для *ed*, который помещает все отличия между файлами файл2 и файл3 в файл файл1 (будьте осторожны!)
-i	Добавить команды *w* (сохранить файл) и *q* (выйти) в конец сценария *ed*
-x	Создание сценария редактора *ed*, который помещает отличия между файлами в файл файл1
-X	То же, что и *-x*, но отличия выделяются
-3	Создает сценарий *ed*, который помещает все различия между файлами файл1 и файл3 в файл1

4.7.6. Команда *grep*

Предположим, что у нас есть какой-то большой файл, и мы хотим найти в нем все упоминания строки *hello*. Сделать это можно так:

```
cat file.txt | grep hello
```

Команда cat file.txt передаст содержимое файла file.txt на стандартный ввод команды *grep*, которая, в свою очередь, выделит строки, содержащие строку *hello*.

4.7.7. Замена символов табуляции пробелами

Команда *expand* заменяет в указанных файлах символы табуляции на соответствующее количество пробелов. Команде можно передать лишь один параметр *-i*, означающий, что замена должна быть только в начале строки.

Формат вызова команды:

```
expand [-i] файлы
```

4.7.8. Форматирование текста

Команда *fmt* форматирует текст, выравнивает его по правой границе и удаляет символы новой строки. Синтаксис вызова команды:

```
fmt [параметры] файлы
```

Параметры команды *fmt* приведены в табл. 4.11.

Таблица 4.11. Параметры команды fmt

Параметр	Описание
-c	Не форматировать первые две строки
-p префикс	Форматировать только строки, начинающиеся с указанного префикса
-s	Не объединять строки
-t	Начинать параграф с красной строки
-w n	Задает максимальную длину строки в *n* символов (по умолчанию 72)

4.7.9. Команды постраничного вывода *more* и *less*

Большой текстовый файл намного удобнее просматривать с помощью команд *less* или *more*. Программа *less* удобнее, чем *more*, если она есть в вашей системе:

```
tac /var/log/messages | grep ppp | less
```

4.7.10. Команды *head* и *tail*: вывод первых и последних строк файла

Команда *head* выводит первые десять строк файла, а *tail* — последние десять. Количество строк может регулироваться с помощью параметра *-n*.

Пример использования:

```
head -n 10 /var/log/messages
tail -n 15 /var/log/messages
```

4.7.11. Команда *split*

Используется для разделения файлов на части. По умолчанию создаются части размером в 1000 строк. Изменить размер можно, указав количество строк, например:

```
split -200 файл1
```

В данном случае файл будет разбит на части по 200 строк в каждой (кроме, возможно, последней части, где может быть меньше строк).

Команду можно также использовать для разделения файлов на части по размеру информации, а не по количеству строк, например с помощью параметра *-b* можно указать количество символов в каждой части. Примеры вызова команды:

```
split -b100b файл
split -b100k файл
split -b100m файл
```

Первая команда разделит файл на части по 100 байтов каждая, вторая — на части по 100 Кбайт каждая, третья — по 100 Мбайт каждая.

4.7.12. Команда *unexpand*

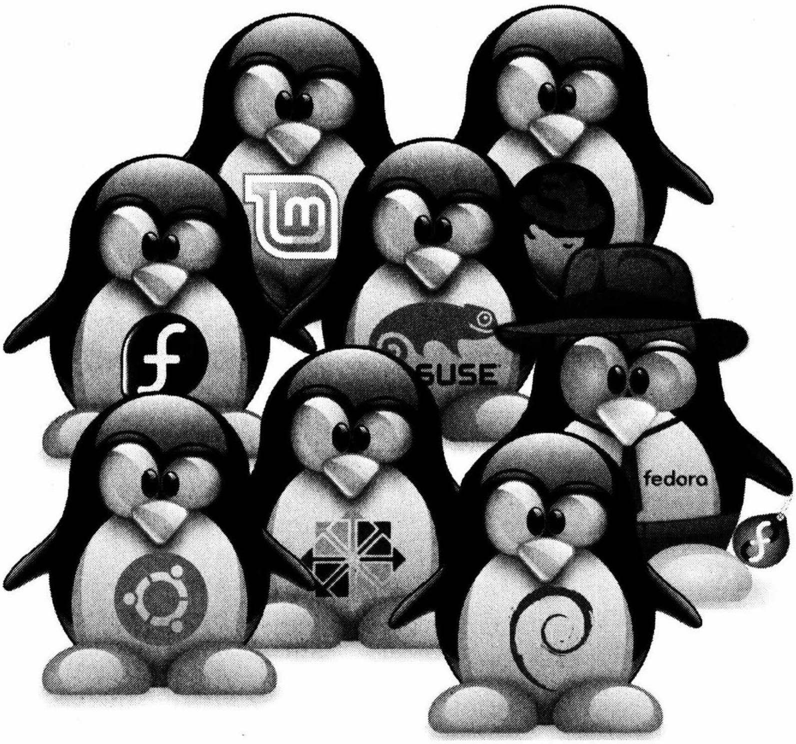
Заменяет последовательные пробелы символами табуляции. По умолчанию 8 пробелов заменяются одним *символом табуляции*. Количество пробелов можно задать с помощью параметра *-t n* (где *n* — количество пробелов).

Синтаксис вызова:

```
unexpand [параметры] файл
```

Глава 5.

Локальная сеть



Следующая часть этой книги начнется с настройки сети. Будет рассмотрено, как настроить локальную Ethernet и беспроводную Wi-Fi-сеть, как установить VPN-соединение. А уже после этого мы рассмотрим установку программного обеспечения, популярные программы и все остальное. Без сети сейчас никак – даже программу не установишь, поэтому приступим.

5.1. Физическая настройка сети Ethernet

Существует много сетевых технологий, но в этой книге мы будем рассматривать настройку локальной сети, построенной на технологии Fast Ethernet. Однако мы рассмотрим ее полностью — от обжатия кабеля до конфигурирования сети в Linux.

Не смотря на наличие уже стандарта Gigabit Ethernet (1000 Мбит/с), стандарт Fast Ethernet (100 Мбит/с) все еще актуален и его скорости вполне хватает для обеспечения работы локальной сети. А стандарт Gigabit Ethernet пока используется в качестве магистрали.

Прежде всего, вам нужно убедиться, что у вас есть сетевой адаптер, поддерживающий технологию FastEthernet. Большинство современных компьютеров оснащены такими сетевыми адаптерами. Как правило, в современных компьютерах и ноутбуках сетевые адаптеры являются интегрированными в материнскую плату и устанавливать их отдельно не нужно.

После это вам нужно подключить сетевой кабель к вашему сетевому адаптеру. Как правило, кабель обжимается администратором сети.

Для создания Fast Ethernet сети вам нужны следующие устройства:

- Сетевые адаптеры — с ними мы уже разобрались;
- Коммутатор (switch) — его можно купить в любом компьютерном магазине. Вместо него сойдет машрутизатор (router), который, как правило, обладает 4-8 портами для подключения локальных компьютеров – этого вполне достаточно для построения небольшой SOHO-сети (Small Office Home Office);
- Витая пара пятой категории — спрашивайте именно такой тип кабеля[1];
- Коннекторы RJ-45 — таких коннекторов вам нужно будет в два раза больше, чем число компьютеров, поскольку кабель нужно будет обжать с двух концов.
- Инструмент для обжимки витой пары — хороший инструмент стоит относительно дорого (примерно как коммутатор), а плохой лучше не покупать. Если не хотите выкладываться, возьмите у кого-нибудь на пару дней.

Теперь приступим к самому процессу обжимки. Внутри кабеля будут 4 витые пары, причем у каждого провода будет своя цветовая маркировка. Суть процесса обжимки заключается в том, чтобы подключить каждый из проводов к нужному контакту коннектора. Сначала нужно поместить провода в коннектор (защищать их необязательно — за вас это сделает инструмент), затем коннектор обратной частью (той, которой он будет вставляться в сетевой адаптер) помещается в инструмент для обжимки и крепко обжимается. Используя приведенную ниже таблицу (табл. 5.1), вы без проблем сможете обжать кабель:

Таблица 5.1. Обжимка витой пары

Контакт	Цвет провода
1	Бело-оранжевый
2	Оранжевый
3	Зелено-белый

1 Для Gigabit Ethernet нужна витая пара 6-ой категории

4	Синий
5	Сине-белый
6	Зеленый
7	Бело-коричневый
8	Коричневый

Обжимать кабель нужно с двух сторон. Один конец подключается к концентратору (или коммутатору), а второй — к сетевому адаптеру. Если вы неправильно (или слабо) обожмете кабель, то ваша сеть работать не будет или же будет работать только на скорости 10 Мбит/с.

Проверить, правильно ли вы обжали кабель очень просто: обратите внимание на коммутатор. Возле каждого порта будет два индикатора. Если горят оба, значит все нормально. Если же горит только один из них, значит, данный порт работает в режиме 10 Мбит/с. А если вообще не горит ни один из индикаторов, значит, вам нужно переобжать кабель.

5.2. Настройка сети с помощью графического конфигуратора

В каждом дистрибутиве Linux есть графические конфигураторы. Конечно, на сервере далеко не всегда устанавливается графический интерфейс, поэтому такие конфигураторы будут недоступны, однако не сказать о них тоже нельзя. Сначала мы рассмотрим графические конфигураторы, а затем рассмотрим настройку сети в консоли – с помощью конфигурационных файлов.

Прежде, чем приступить к настройке сети, ради справедливости нужно отметить, что в большинстве случаев настраивать ничего не придется – во всех современных сетях есть DHCP-сервер, который и настраивает все остальные компьютеры. Настройка сети может понадобиться разве что на самом DHCP-сервере, но это только в том случае, когда вы настраиваете сеть с нуля. Опять-таки, если вы подключили свои домашние компьютеры к маршрутизатору, то в качестве DHCP-сервера будет выступать сам маршрутизатор и вам в большинстве случаев вообще не придется ничего настраивать.

В Ubuntu для настройки локальной сети щелкните по значку сети в правом верхнем углу (рис. 5.1). Выберите команду **Параметры соединения**, чтобы открыть конфигуратор сети.

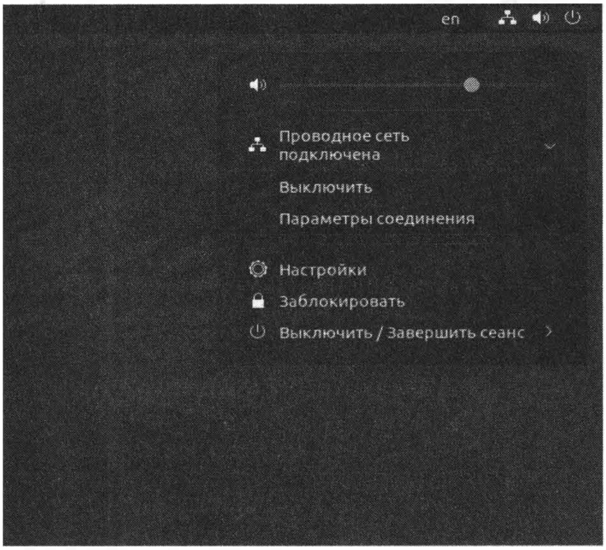


Рис. 5.1. Системное меню

В появившемся окне **Сеть** нажмите кнопку с изображением шестеренки напротив строки **Подключено** – 1000 Мбит/с (рис. 5.2).

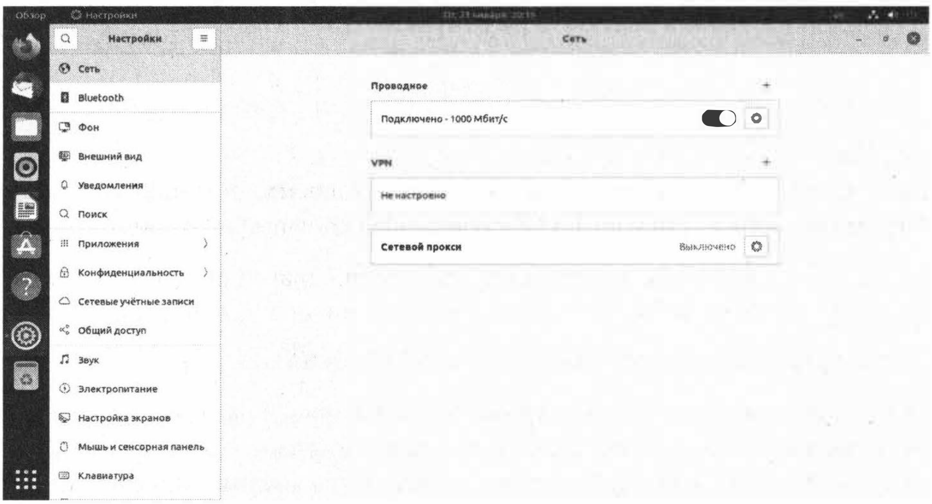


Рис. 5.2. Окно Сеть

В появившемся окне на вкладке **Сведения о системе** убедитесь, что включен флажок **Подключаться автоматически**, чтобы соединение устанавливалось автоматически при загрузке системы (или при входе в систему, если выключен флажок **Сделать доступным для других пользователей**).

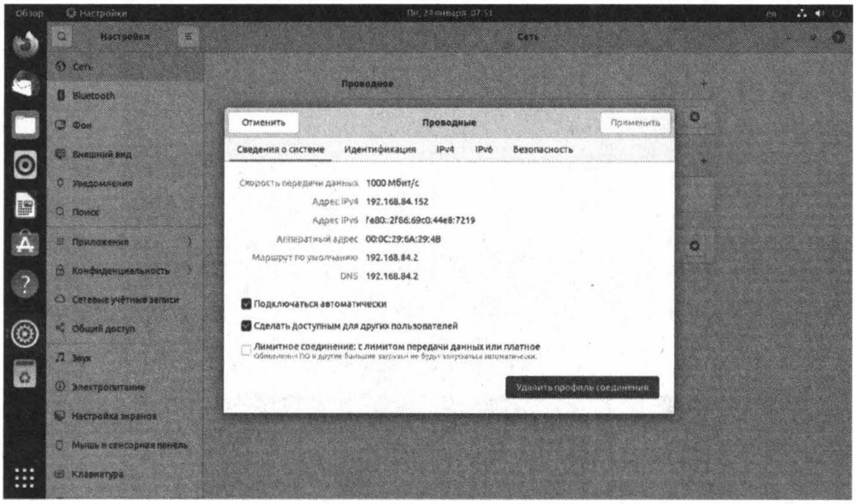


Рис. 5.3. Сведения о системе

Далее перейдите на вкладку IPv4 для изменения параметров протокола IP. По умолчанию сеть настроена на использование протокола DHCP (рис. 5.4). Для ручной настройки выберите **Вручную** и укажите (рис. 5.5):

- IP-адрес, маску сети, IP-адрес шлюза (все эти параметры можно получить у администратора сети).
- IP-адреса DNS-серверов, рекомендуется использовать IP-адреса Google. При указании несколько IP-адресов разделять их нужно запятыми, как показано на рис. 5.5.

Для сохранения настроек нажмите кнопку **Применить.**

В Astra Linux настройка осуществляется аналогично, только интерфейс конфигуратора немного другой. Щелкните правой кнопкой мыши на значке сетевого соединения и выберите команду **Изменить соединения** (рис. 5.6). В появившемся окне выберите сетевое соединение и нажмите кнопку с изображением *шестеренки* (рис. 5.7).

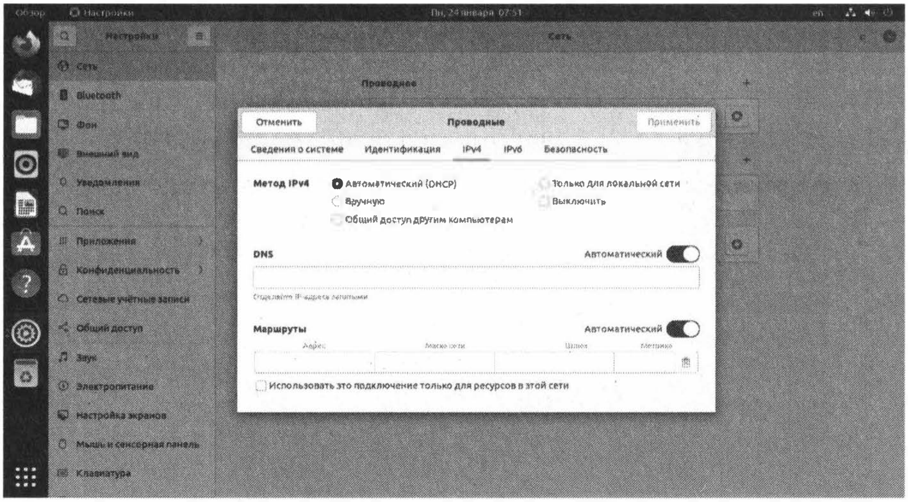


Рис. 5.4. Используем DHCP

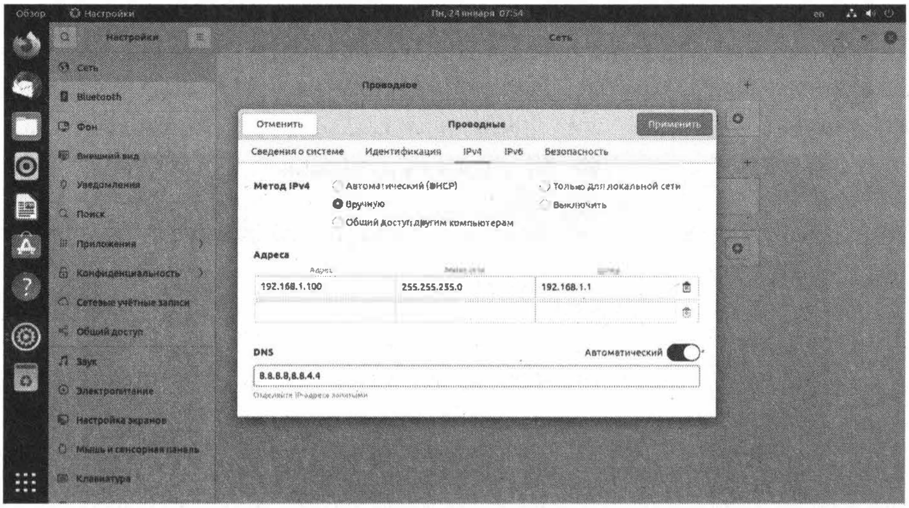


Рис. 5.5. Ручная настройка

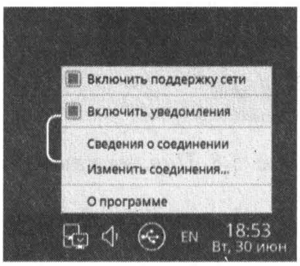


Рис. 5.6. Вызов конфигуратора сети в Astra Linux

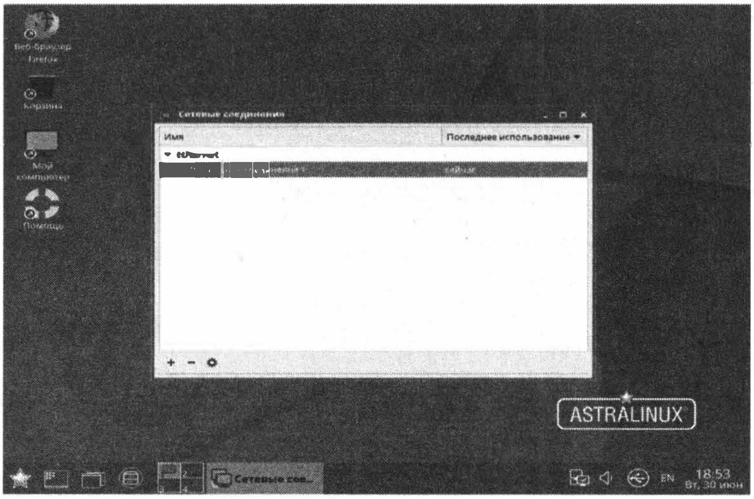


Рис. 5.7. Конфигуратор сети

Далее нужно перейти на вкладку **Параметры IPv4** и установить параметры сети, как это было сделано ранее в Ubuntu. IP-адреса DNS-серверов также указываются через запятые. Для применения настроек нажмите кнопку **Сохранить**.

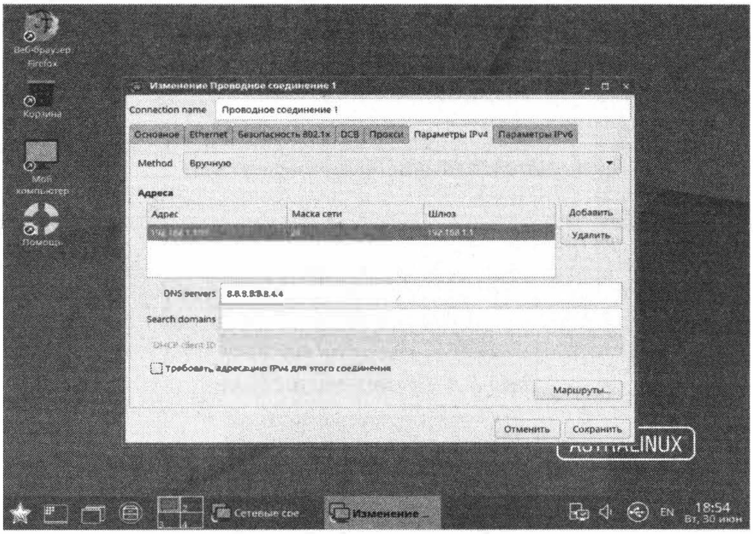


Рис. 5.8. Параметры IPv4

В 99% случаев вам не нужно редактировать параметры Ethernet-соединения, поскольку настройка данного соединения осуществляется по протоколу

DHCP автоматически. Конечно, в некоторых случаях, например, когда вам нужно по тем или иным причинам клонировать MAC-адрес устройства или у вас нет DHCP-сервера и нужно определить конфигурацию интерфейса вручную, придется редактировать параметры проводного соединения. Также настройка интерфейса вручную может потребоваться на компьютере, который используется в качестве DHCP-сервере.

На вкладке **Ethernet** можно (рис. 5.9):

- Выбрать физическую сетевую плату, которая будет использоваться для этого соединения (список **Device**).
- Изменить MAC-адрес соединения (**Cloned MAC address**).
- Задать MTU соединения.

5.3. Команда *ifconfig*

Команда *ifconfig* используется для настройки и отображения параметров сетевого интерфейса. Если ввести команду *ifconfig* без параметров, то мы получим список всех активных интерфейсов, например:

```
ens33     Link encap:Ethernet  HWaddr 00:0C:29:C2:0D:D1
          inet addr:192.168.52.154  Bcast:192.168.52.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fec2:dd1/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:425 errors:0 dropped:0 overruns:0 frame:0
          TX packets:406 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:41540 (40.5 Kb)  TX bytes:39618 (38.6 Kb)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:96 errors:0 dropped:0 overruns:0 frame:0
          TX packets:96 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:5760 (5.6 Kb)  TX bytes:5760 (5.6 Kb)
```

Примечание. Даже если сетевые интерфейсы не настроены, в выводе ***ifconfig*** должен быть интерфейс *lo*. Если его нет, служба *network* не запущена. Для ее запуска введите команду от пользователя root: `service network start`.

Разберемся, что есть что. У нас есть два сетевых интерфейса – **ens33** (Ethrenet-адаптер) и **lo** (интерфейс обратной петли). Второй используется для тестирования работы сети и всегда имеет IP-адрес 127.0.0.1. Также он может использоваться для обращения к локальному компьютеру. Например, если на вашем компьютере установлен MySQL-сервер, то в PHP-сценариях можно смело указывать адрес 127.0.0.1 или localhost, чтобы указать, что вы обращаетесь к MySQL-серверу, запущенному на этом компьютере.

Рассмотрим поля вывода *ifconfig*:

- HWaddr – содержит аппаратный MAC-адрес (в нашем случае это 00:0C:29:C2:0D:D1);
- inet addr – содержит IPv4-адрес интерфейса (192.168.52.154);
- inet6 addr – содержит IPv6-адрес интерфейса;
- Bcast – адрес для широковещательной передачи;
- Mask – маска сети;
- MTU – значение MTU (Maximum transmission unit);
- collisions – счетчик коллизий, если количество коллизий больше 0, с вашей сетью творится что-то неладное. Как правило, в современных коммутируемых сетях коллизии отсутствуют вообще.
- RX packets – количество принятых пакетов;
- TX packets – количество переданных пакетов;
- RX bytes – количество принятых байтов;
- TX bytes – количество переданных байтов.

Получить список всех интерфейсов, а не только активных, можно с помощью параметра *-a*:

```
sudo ifconfig -a
```

С помощью команды *ifconfig* можно *деактивировать* (down) и *активировать* (up) любой интерфейс:

```
sudo ifconfig ens33 down
sudo ifconfig ens33 up
```

Иногда такой "перезапуск" интерфейса помогает наладить его работу. Особенно это полезно для "подвисших" беспроводных интерфейсов (wlan*).

С помощью *ifconfig* можно назначить IP-адрес и другие сетевые параметры интерфейса, например:

```
sudo ifconfig ens33 down
sudo ifconfig ens33 192.168.1.1 up
```

Сначала мы деактивируем интерфейс, затем назначаем новый IP-адрес и "поднимаем" (параметр *up*) интерфейс. Можно эту последовательность действий выполнить и за три команды:

```
sudo ifconfig ens33 down
sudo ifconfig ens33 192.168.1.1
sudo ifconfig ens33 up
```

Аналогичным образом можно назначить маску сети и широковещательный адрес:

```
sudo ifconfig eth0 10.0.0.1 netmask 255.255.255.240 broadcast 10.0.0.15
```

После изменений параметров интерфейса не забудьте его активировать с помощью команды *up*:

```
sudo ifconfig eth0 up
```

Изменить MTU интерфейса можно так:

```
sudo ifconfig eth0 down
sudo ifconfig eth0 mtu XXXX up
```

Самой интересной возможностью *ifconfig* является перевод интерфейса в так называемый *promiscuous mode*. В обычном режиме сетевая карта, если принимает пакет, который был адресован не ей, она его просто отбрасывает. В *promiscuous mode* карта будет принимать пакеты, даже которые не были ей адресованы. Такая возможность может понадобиться для перехвата трафика:

```
sudo ifconfig eth0 promisc
```

Перевести карту в обычный режим можно командой:

```
sudo ifconfig eth0 -promisc
```

Помните, что параметры, заданные с помощью *ifconfig*, хранятся только до следующей перезагрузки. Если вы хотите сохранить назначенные параметры, нужно или использовать конфигураторы сети (описаны ранее) или редактировать файлы конфигурации (будут описаны далее).

5.4. Имена сетевых интерфейсов в Linux

В Linux у каждого сетевого интерфейса есть свое имя, чтобы администратор сразу понимал, что это за соединение. Например, у всех PPP-соединений имя ppp?, где ? – номер соединения: первое соединение называется ppp0, второе – ppp1 и т.д. Нумерация соединений начинается с нуля. В таблице 5.2 приведены названия часто используемых сетевых интерфейсов.

Таблица 5.2. Имена сетевых интерфейсов в Linux

Название интерфейса	Описание
lo	Интерфейс обратной петли
eth, он же ens	Сетевой интерфейс Ethernet
ppp	Соединение PPP (Point-to-Point)
wlan	Беспроводной сетевой интерфейс
tr	Сетевой интерфейс Token Ring
plip	Сетевой интерфейс PLIP (Perallel Line IP). Очень старый интерфейс, вряд ли вы будете с ним иметь дело
sl	Сетевой интерфейс SLIP (Serial Line IP). Тоже "древний" интерфейс
fddi	Интерфейс к карте FDDI
ax	Сетевой интерфейс любительского радио AX.25

Интерфейсы создаются автоматически ядром для каждого обнаруженного сетевого устройства. Исключения составляют разве что интерфейсы **ppp**, которые создаются во время настройки сети. Например, у вас может быть PPPoE-соединение к провайдеру и сетевая карта *eth0*. Сетевой интерфейс *eht0* для сетевой карты будет создан автоматически ядро. Но ядро не знает, как вы будете использовать эту карту – или для подключения к локальной сети или как PPPoE-соединение с провайдером, поэтому оно не может создать интерфейс ppp0, прочитав ваши мысли.

Все было бы хорошо, если по непонятным причинам в последних версиях ядра интерфейсы не переименовали. Так, старый добрый *eth0* теперь почему-то называется *ens33*, в некоторых дистрибутивах даже *enp3s0*. Логики в таких именах не очень много, да и хочется, чтобы имена интерфейсов были такие же, как раньше – как-никак 20-летняя традиция. Ради справедливости нужно отметить, что в Astra Linux интерфейс называется как нужно – eth0 (рис. 5.9).

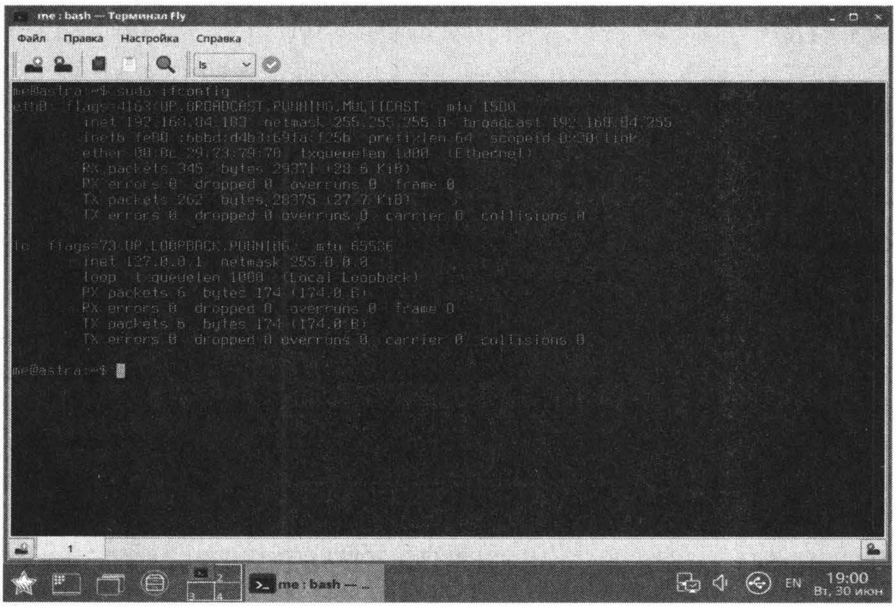


Рис. 5.9. Вывод ifconfig в Astra Linux

На самом деле, логика в новых названиях есть, но только с точки зрения самого компьютера и разработчиков udev. Пользователям и администраторам такая логика не очень понятна. Например, имена, вовлекающие предоставленный BIOS индекс для встроенных в материнскую плату устройств (ID_NET_NAME_ONBOARD) выглядят так – *eno1*. Имена адаптеров, под-

ключающихся к шине PCI Express, будут содержать номер слота (ID_NET_NAME_SLOT), например, *ens1*. Имена, вовлекающие физическое/географическое расположение коннектора (ID_NET_NAME_PATH), выглядят еще сложнее – enp2s0. Ну и самые "страшные" – это имена с MAC-адресами интерфейсов (ID_NET_NAME_MAC). Пример такого имени: *enx78e7d1ea46da*.

Конечно, можно оставить все как есть – все будет прекрасно работать. Но если вам привычнее обычные имена, тогда есть несколько способов вернуть все, как было раньше.

Первый способ заключается в редактировании файла /etc/udev/rules.d/70-my-network.rules (или 70-persistent-net.rules). Если такого файла у вас нет, значит, его нужно создать. Примерное содержимое файла следующее:

```
# PCI device (r8169)
SUBSYSTEM=="net", ACTION=="add", ATTR{address}=="xx:xx:xx:xx:xx:xx",
NAME="eth0"

# USB device (usb)
SUBSYSTEM=="net", ACTION=="add", ATTR{address}=="yy:yy:yy:yy:yy:yy",
NAME="wlan0"
```

Здесь *xx* нужно заменить MAC-адресом вашей сетевой карты, а *yy* – на MAC-адрес вашего беспроводного адаптера.

Есть и второй способ – передать ядру параметр *net.ifnames=0*.

5.5. Общие конфигурационные файлы

Наверняка вам интересно, куда конфигураторы сохраняют введенные вами настройки? Об этом мы обязательно поговорим, тем более, что файлы конфигурации сети отличаются в зависимости от используемого дистрибутива. Но сначала мы рассмотрим общие файлы конфигурации, имеющие отношение к сети. Эти файлы есть во всех дистрибутивах Linux и их назначение одинаково.

Файл /etc/hosts

Когда еще не было системы разрешения доменных имен (DNS) в этом файле прописывались IP-адреса узлов и соответствующие им имена узлов. Эти файлы тиражировались от компьютера к компьютеру. Если в сеть добавлялся новый компьютер, изменения нужно было внести в файлы /etc/hosts на всех компьютера. Сегодня этот файл практически не используется и в нем содержится только IP-адрес узла localhost – 127.0.0.1 и аналогичные IPv6-адреса. Формат записей этого файла следующий:

```
IP-адрес    Имя
```

Например:

```
127.0.0.1   localhost
```

Файлы /etc/hosts.allow и /etc/hosts.deny

Содержит IP-адреса узлов, которым разрешен (hosts.allow) или запрещен (hosts.deny) доступ к сервисам данного узла. Опять таки – это устаревшие конфигурационные файлы. Вы можете их использовать, однако в настоящее время ограничения доступа к определенной службе или узлу достигается путем редактирования правил брандмауэра, а не путем редактирования этих файлов. При желании, конечно, вы можете их использовать, но среди своих коллег вы будете выглядеть настоящим динозавром.

Файл /etc/host.conf

Содержит параметры разрешения доменных имен, который указывается директивой *order*. Так, `order hosts, bind` означает, что сначала поиск IP-адреса по доменному имени будет произведен в файле /etc/hosts, а затем – с помощью DNS-сервера. Если для вас это существенно, можете изменить порядок на `bind, hosts`, ведь файл hosts уже никем давно не используется и искать там нечего.

Директива `multi on` в этом файле означает, что одному доменному имени могут соответствовать несколько IP-адресов.

Файл /etc/hostname

Обычно в этом файле содержится имя узла. Оно записано одной строкой, больше никаких директив в этом файле нет.

Файл /etc/motd

Содержит сообщение дня, которое может использоваться некоторыми сетевыми службами (например, FTP) и отображаться при входе пользователя в систему. Чтобы ваша сетевая служба использовала именно этот файл, нужно проверить ее файл конфигурации.

Файл /etc/resolv.conf

Обычно в этом файле содержатся параметры резолвера доменных имен. Директива *search* задает список доменов, в которых будет произведен поиск доменного имени, если оно введено не полностью. Например, пусть директива *search* определена так:

```
search org org.ua net
```

Вы вводите доменное имя не полностью (например, в браузере) – *dkws*. Тогда резолвер попытается сначала разрешить доменное имя dkws.org, потом – dkws.org.ua, а уж потом, если ни один из первых двух вариантов не был разрешен в IP-адрес – dkws.net.

Директива *nameserver* позволяет определить IP-адрес сервера DNS. В одной директиве можно указать только один сервер. Всего в файле конфигурации может быть определено до четырех серверов имен:

```
nameserver 8.8.8.8
nameserver 8.8.4.4
```

Пример файла конфигурации /etc/resolv.conf приведен в листинге 5.1.

Листинг 5.1. Файл /etc/resolv.conf

```
search dkws.org.ua

nameserver 8.8.8.8
nameserver 8.8.4.4
```

Файл /etc/services

Все мы знаем, что за сетевыми службами закрепляются определенные номера портов. Например, службе *pop3* соответствует порт 110, *ftp* – порт 21. Эти соответствия и определяются в файле /etc/services. Обычно вам не придется редактировать этот файл. Скорее, вы будете его использовать в качестве информационного, чтобы знать, какой порт соответствует какой службе.

Файл /etc/protocols

Тоже сугубо информационный файл (его можно редактировать, но вряд ли вы будете это делать). В нем описываются имена протоколов, их номера, псевдонимы и комментарии, а также ссылки на RFC-документы. Полезный файл, если вы собрались разобраться по очереди во всех сетевых протоколах.

Файл /etc/network/interfaces: конфигурация сети в Astra Linux

Основной конфигурационный файл сети в Astra Linux называется /etc/network/interfaces. Он содержит параметры сетевых интерфейсов, если не используется Network Manager. Вот пример описания интерфейса *eth1* со статическим IP-адресом:

```
iface eth1 inet static
   address 192.168.1.2                 # IP-адрес
   netmask 255.255.255.0               # маска сети
   gateway 192.168.1.1                 # Шлюз
```

Думаю, все ясно. Если интерфейс настраивается по DHCP, его конфигурация будет выглядеть так:

```
auto eth0
iface eth0 inet dhcp
```

Каталог /etc/NetworkManager/system-connections: конфигурация сети в Ubuntu

В Ubuntu параметры сетевого интерфейса хранятся в каталоге /etc/NetworkManager/system-connections. В этом каталоге находятся несколько файлов, имена этих файлов соответствуют названиям соединений в NetworkManager. Например, в файле 'Проводное соединение 1.nmconnection' находятся параметры первого проводного соединения (лист. 5.2).

Листинг 5.2. Пример настройки интерфейса со статическим IP-адресом

```
[802-3-ethernet]
duplex=full

[connection]
id=Проводное соединение 1
uuid=00779167-0a51-44ad-a2b3-b5765a65b496
type=802-3-ethernet
timestamp=1419337229

[ipv6]
method=auto
ip6-privacy=2

[ipv4]
method=manual
dns=8.8.8.8;
addresses1=192.168.1.101;24;192.168.1.1;
```

Как видите, протокол IPv4 настраивается вручную (method=manual). Был задан DNS-сервер (8.8.8.8), а также IP-адрес узла (192.168.1.101), маска сети

(24 – соответствует 255.255.255.0), а также адрес шлюза – 192.168.1.1. Если нужно добавить еще один IP-адрес, просто добавьте строку addressesN:

```
addresses2=192.168.1.102;24;192.168.1.1;
```

Для настройки интерфейса по DHCP измените параметр *method*:

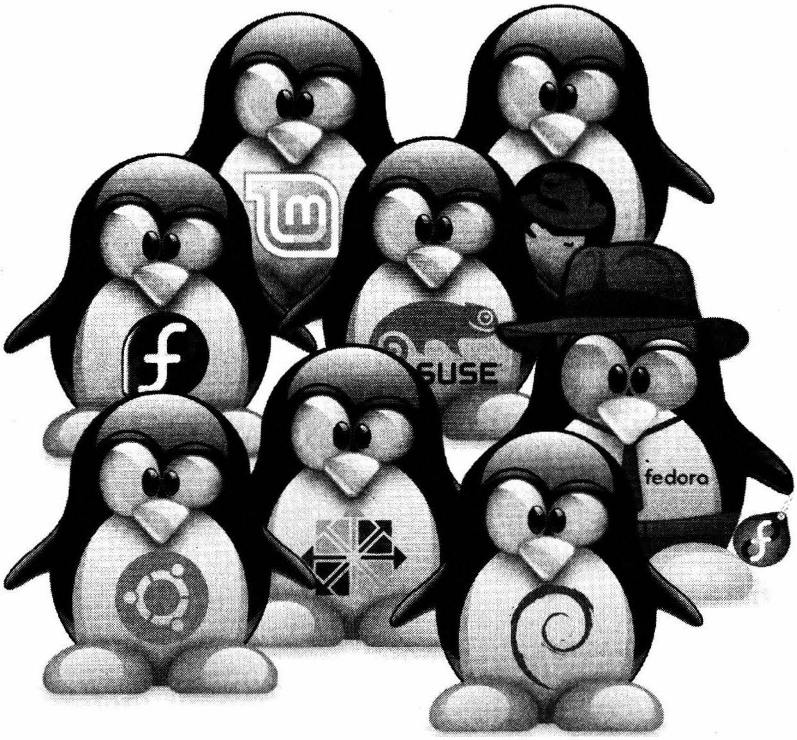
```
[ipv4]
method=auto
```

Если вы изменили параметры интерфейса, контролируемого Network Manager и не через интерфейс NM, а вручную, тогда нужно перезапустить Network Manager:

```
# service network-manager restart
```

Глава 6.

Беспроводная сеть Wi-Fi



Беспроводные сети быстро завоевали популярность. Оно и понятно – быстро, просто, дешево. Судите сами: не нужно прокладывать кабели, обжимать их. Все, что нужно – купить «коробочку» (самые дешевые сейчас можно встретить чуть дороже 1000 рублей) и включить ее. Можно даже не настраивать, а использовать те пароли, которые указаны на самой «коробочке». В итоге, начальное «разворачивание» сети у вас займет 5 минут – на выбор и размещение WiFi-маршрутизатора и его включение, и еще по 1 минуте на настройку каждого узла, который будет к подключаться к беспроводной сети. Это потом уже вам захочется улучшить прием, выбрав другой канал, поменять пароли по умолчанию и т.д. Если у вас к беспроводной сети подключается 5 устройств, то на первоначальную настройку всей сети будет потрачено 10-15 минут.

Давно прошли те времена, когда настройка беспроводной сети в Linux напоминала шаманские заклинания и пляски с бубном вокруг компьютера. Сейчас все происходит так же, как и в случае с «нормальными» операционными системами – выбрал сеть, ввел пароль и работаешь.

Мы рассмотрим, как подключиться к Wi-Fi с помощью графического интерфейса, и как проделать то же самое в консоли. Такой «лайфхак» может

понадобиться вам при администрировании сервера без графического интерфейса. Обычно серверы подключают посредством проводной сети, что надежнее, но и с проводной сетью может что-то случиться, а какой-никакой канал нужно обеспечить, поэтому в качестве резервного канала можно использовать беспроводную сеть. Вот ее настройку мы и рассмотрим.

6.1. Настройка беспроводной сети с помощью графического интерфейса

Мы будем считать, что беспроводной адаптер подключен к компьютеру и включен. Если у вас стационарный компьютер, убедитесь, что вы подключили WiFi-адаптер к USB-порту. Также убедитесь, что он работает – на самих адаптерах иногда есть переключатели, позволяющие включать/выключать их. Если адаптер работает, индикатор на нем светится.

С ноутбуками может быть все немного сложнее. Адаптер, как правило, встроен в ноутбук, но может быть выключен. Обычно включение/выключение адаптера происходит посредством комбинации клавиш Fn + Fx, где Fn – это клавиша Fn (находится, как правило, справа от левого Ctrl), а Fx – одна из функциональных клавиш (F1 – F12). Над такой клавишей обычно есть изображение беспроводной сети или самолета. Так, на ноутбуках HP на клавише F12 есть изображение самолета, нажатие комбинации Fn + F12 переводит ноутбук в режим полета, когда Wi-Fi выключается, повторное нажатие переключает ноутбук в обычный режим – в нем Wi-Fi доступен. На некоторых ноутбуках могут быть аппаратные переключатели (выполненные в виде переключателя или кнопки) для включения/выключения WiFi. Если не разберетесь, обратитесь к документации по ноутбуку.

Итак, если беспроводной адаптер подключен и включен, в меню NetworkManager появится возможность подключения к беспроводной сети (рис. 6.1). Если в меню не появился пункт **Wi-Fi сеть не подключена**, значит, есть какая-то проблема с беспроводным адаптером. Как правило, он просто выключен. Если данный пункт в меню появился, выберите его.

Далее нужно выбрать команду **Выбрать сеть** (рис. 6.2). Появится список сетей, в котором нужно выбрать сеть и нажать кнопку **Соединиться** (рис. 6.3). Далее введите пароль и дождитесь подключения к сети.

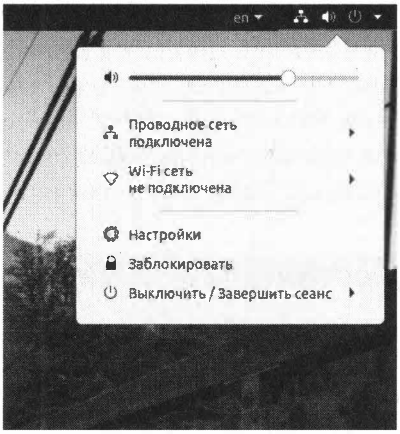


Рис. 6.1. Wi-Fi активирован, но не подключен к сети

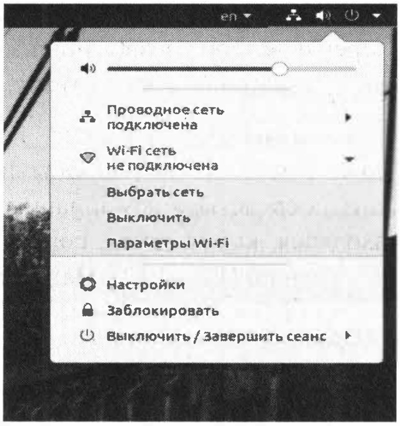


Рис. 6.2. Используйте команду Выбрать сеть

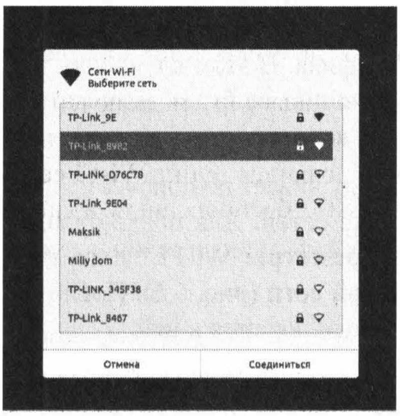


Рис. 6.3. Список сетей

Команда **Параметры Wi-Fi** открывает окно, в котором будет список сетей. Нажмите значок **шестеренки** напротив сети, к которой вы подключены, чтобы открыть ее параметры. Особо интересного в этих параметрах нет, разве что вы можете узнать аппаратный адрес (MAC-адрес) беспроводного адаптера, а также разрешить/запретить подключаться к сети автоматически и сделать это подключение доступным для других пользователей компьютера (рис. 6.4)

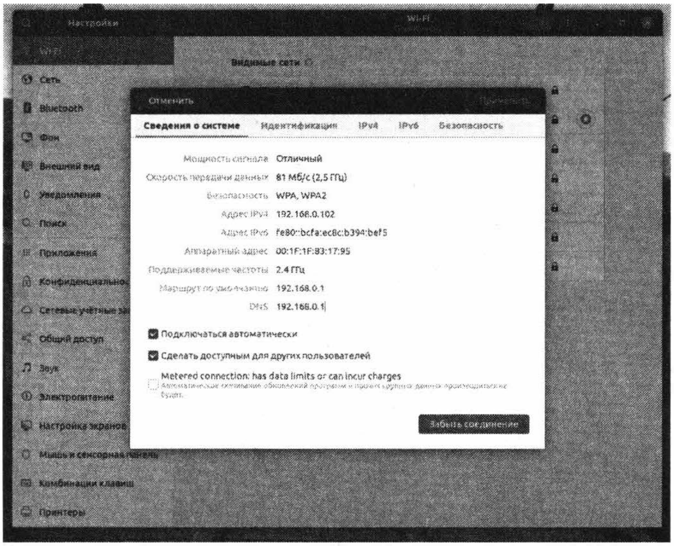


Рис. 6.4. Параметры беспроводного соединения

Параметр **Metered connection: has data limits or can incur charges** полезен, если вы подключаетесь к Wi-Fi, созданной путем «расшаривания» соединения на мобильном телефоне или же при подключении к платной Wi-Fi отеля, где есть тарификация трафика. В этом случае система при подключении по тарифицируемому соединению не будет загружать обновление программ и прочих больших объемов данных.

Отдельного внимания заслуживает подключение к скрытой сети. Такие сети не отображаются в списке сетей. Для подключения к такой сети в Ubuntu нужно открыть окно **Параметры Wi-Fi**, далее из меню выбрать команду **Подключиться к скрытой сети** (рис. 6.5). После этого нужно будет ввести SSID сети и пароль для подключения к ней.

В Astra Linux подключение к Wi-Fi еще удобнее, чем в Ubuntu. Хотя бы потому, что не нужно щелкать по пункту **Wi-Fi сеть не подключена** и не нужно

выбирать команду **Выбрать сеть** – вы экономите целых два клика мышки. Щелкните по значку NetworkManager и вы сразу получите список сетей.

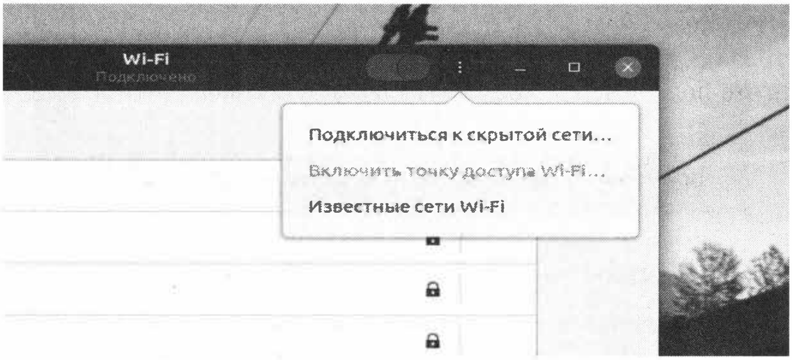


Рис. 6.5. Подключение к скрытой сети

Если нужно сети нет в списке, используйте меню **Еще сети**, а для подключения к скрытой сети – команду **Подключиться к скрытой сети Wi-Fi**.

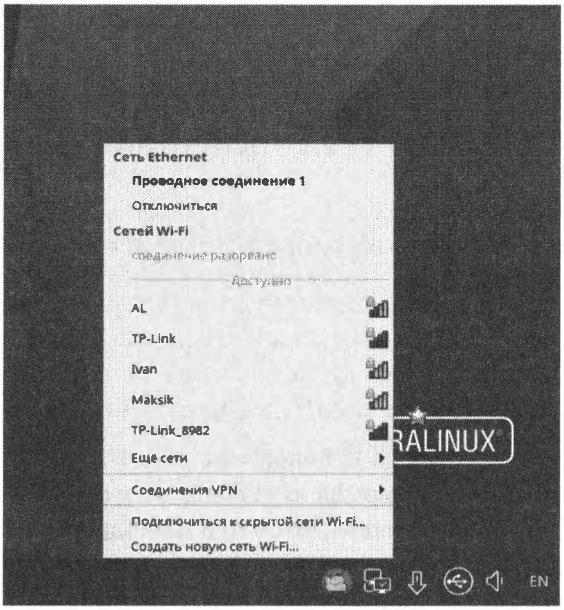


Рис. 6.6. Выбор сети

Далее нужно ввести только пароль для подключения к сети и дождаться самого подключения.

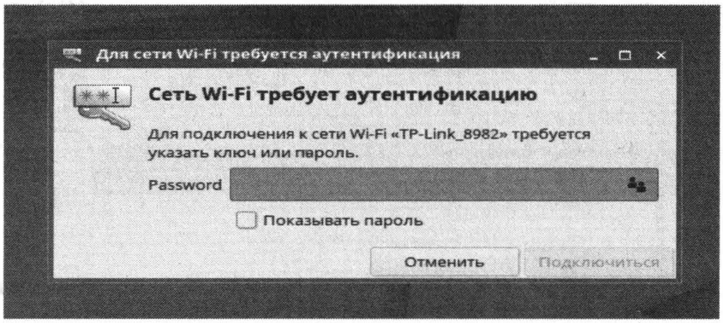


Рис. 6.7. Ввод пароля

Примечание. Обычно WiFi-сети требуют пароль для подключения к ним. Такие сети в списке сетей отмечаются замочком. Если замочка нет, то сеть считается доступной для всех. Если вы не знаете, что это за сеть, подключаться к ней не рекомендуется, поскольку часто открытые сети разворачиваются злоумышленниками для кражи ваших данных, которые вы будете передавать через их сеть.

6.2. Настройка беспроводного соединения в командной строке (WEP-шифрование)

Первое, что нужно сделать – посмотреть, какие сетевые адаптеры имеются у нас на компьютере:

```
sudo ifconfig -a
```

Вывод будет содержать имена и подробное описание всех сетевых интерфейсов, которые удалось обнаружить утилите *ifconfig*. Если не был обнаружен желаемый, то причина заключается только в одном – нет драйверов для него и не включена поддержка этого интерфейса в ядре Linux. Обычно беспроводной интерфейс называется wlan0. Запустим его:

```
sudo ifconfig wlan0 up
```

Опция *up* говорит команде *ifconfig* запустить для работы («поднять») сетевое устройство.

Теперь нам надо просканировать эфир вокруг себя на наличие доступных беспроводных сетей:

```
sudo iwlist wlan0 scan
```

Здесь *wlan0* – имя нашего адаптера, *scan* – режим сканирования сетей.

Результатом работы *iwlist* будет детальный отчет, из которого на данном этапе нас интересует только одна строчка: ESSID:»Some_Name». Значение параметра ESSID («Some_Name») – это имя беспроводной точки доступа. Теперь мы знаем, к какой конкретно Wifi-сети мы будем подключаться.

Выполняем подключение:

```
sudo iwconfig wlan0 essid Имя_сети key Пароль
```

Здесь все просто: мы указываем имя адаптера, SSID сети и пароль для подключения к ней.

Команда *iwconfig* по умолчанию использует для ключа шифрования данные в шестнадцатеричном виде HEX. Если вы хотите указать ключ в виде простого текста (ASCII), вам необходимо использовать опцию *s*.

Например, так:

```
sudo iwconfig wlan0 essid Some_Name key s:Wireless_Key
```

Соединение установлено.

Последний шаг – получаем от dhcp-сервера Wifi-точки IP-адрес:

```
sudo dhclient wlan0
```

Естественно, вышеуказанные шаги выполнять каждый раз утомительно. Можно упростить процесс установки соединения, написав скрипт подключения, в котором мы объединим все эти команды в одно целое:

```
#!/bin/bash
ifconfig wlan0 up
iwconfig wlan0 essid Some_Name key s:Wireless_Key
sleep 10
dhclient wlan0
```

Здесь мы добавили еще одну команду *sleep* с параметром *10 секунд*. Это рекомендуется делать перед получением IP-адреса для надежности установки соединения.

Сохраняем этот файл под каким-либо именем (например, wireless_up) и делаем его исполняемым командой:

```
sudo chmod u+x wireless_up
```

Переносим wireless_up по пути /usr/local/bin, чтобы сделать его глобально видимым всей системой.

Теперь вам достаточно набрать в командной строке:

```
sudo wireless_up
```

... и соединение будет установлено.

6.3. Соединение с точкой доступа по WPA-шифрованию

Соединение с таким шифрованием поддерживает только утилита wpa_supplicant, поэтому она нам понадобится. Также, опять-таки, предполагаем, что мы знаем ключ (пароль) шифрования этой точки доступа.

Генерируем пароль на основе этого ключа с помощью утилиты wpa_passphrase, которая входит в состав пакета wpa_supplicant. Дело в том, что пароль, который мы будем использовать далее, должен быть в виде шестнадцатеричного числа:

```
sudo wpa_passphrase ssid password
```

Утилита выдаст сгенерированную строку *psk*, которую мы вставим в конфигурационный файл wpa_supplicant.conf:

```
Network={
ssid=SSID
psk=PSK }
```

Это очень упрощенный файл конфигурации, но он будет работать. Возможно, вам потребуется добавить в шапку этого файла еще одну строку:

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=wheel
```

для предоставления необходимых прав доступа.

"Поднимаем" интерфейс wlan0:

```
sudo ifconfig wlan0 up
```

Указываем, к какой точке мы хотим подключиться:

```
sudo iwconfig wlan0 essid ssid
```

Запускаем утилиту wpa_supplicant на установку соединения:

```
sudo wpa_supplicant -B -Dwext -i wlan0 -c /etc/wpa_supplicant.conf
```

Разберемся, какие параметры мы указали:

- -B – запускать команду wpa_supplicant в фоновом режиме;
- -Dwext – говорим утилите wpa_supplicant использовать драйвер wext для интерфейса wlan0;
- -i – задаем настраиваемый сетевой интерфейс (wlan0 в нашем случае);
- -c – указываем путь к конфигурационному файлу wpa_supplicant.conf.

Проверяем, что соединение установлено:

```
sudo iwconfig wlan0
```

На выводе увидим подробную информацию об интерфейсе wlan0.

Получаем локальный IP-адрес:

```
sudo dhclient wlan0
```

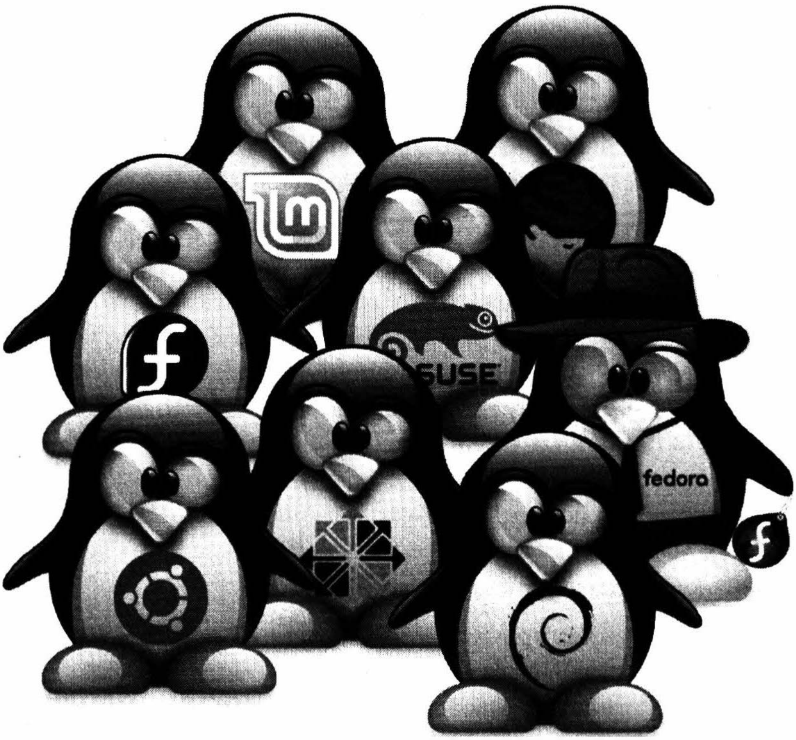
Упрощаем процесс, создав в файле /etc/network/interfaces запись следующего вида:

```
auto wlan0
iface wlan0 inet dhcp
pre-up wpa_supplicant -Bw -Dwext -i wlan0 -c /etc/wpa_supplicant.conf
post-down killall -q wpa_supplicant
```

В зависимости от дистрибутива Linux, существует множество способов настройки Wifi-соединений. Приведенные примеры позволяют настроить соединение практически в любой Linux-системе. Главное, чтобы сам беспроводной адаптер поддерживался в Linux на уровне драйверов.

Глава 7.

VPN-соединение



7.1. Зачем нужна VPN

VPN (*Virtual Private Network*) – очень популярная в последнее время технология. Технология довольно многогранная и может использоваться для самых разных целей. Основная ее задача – шифрование трафика, передаваемого по VPN-соединению. Пользователи используют VPN для достижения одной из целей – обеспечение анонимности и/или обход блокировки ресурса, получение безопасного доступа к сети предприятия из небезопасной сети.

Бывает так, что контролирующие органы закрывают доступ к тем или иным ресурсам, которые пользователь хочет получить. Законность и моральную сторону этих вопросов рассматривать не будем, тем более что иногда ресурсы закрываются по политическим причинам. Вовсе не означает, что если ресурс заблокирован, то сразу на нем продают наркотики, оружие и детскую порнографию. В различных странах запретили доступ к популярным социальным сетям сугубо по политическим причинам и тем самым способствовали совершенствованию технических навыков большей части населения – даже обычная домохозяйка знает, что такое VPN и как его использовать для подключения к Одноклассникам.

Вторая распространенная причина – подключение к офисной сети для передачи документов, составляющих коммерческую тайну. Поскольку не всегда сеть, через которую проходит соединение с офисом, безопасна (например, это может быть сеть отеля, ресторана, аэропорта), то доверять такому соединению нельзя и поэтому настраивают передачу данных по зашифрованному VPN-соединению. Так можно быть уверенным, что важные документы не будут перехвачены.

7.2. Настройка VPN-подключения в Ubuntu

Для настройки подключения к виртуальной частной сети в Ubuntu выполните следующие действия:

1. Откройте окно **Настройки**
2. Перейдите в раздел **Сеть**
3. В группе **VPN** нажмите кнопку **+**
4. Выберите тип протокола – OpenVPN или PPTP. Параметры доступа нужно узнать у администратора VPN-сервера или в службе поддержки VPN-сервиса. Можно также выбрать вариант **Импортировать из файла** и выбрать файл, созданный нами при настройке учетной записи пользователя. Тогда можно пропустить указанные далее действия
5. Выберите сертификаты, сгенерированные в процессе настройки сервера и создания учетной записи пользователя, введите пароль от сертификата.
6. Введите IP-адрес или доменное имя сервера VPN. Остальные параметры можно не изменять
7. Если нужно изменить порт соединения, нажмите кнопку **Дополнительно** и укажите порт и/или другие дополнительные параметры
8. Нажмите кнопку **Добавить** в верхнем правом углу
9. Щелкните по созданному соединению для установки подключения

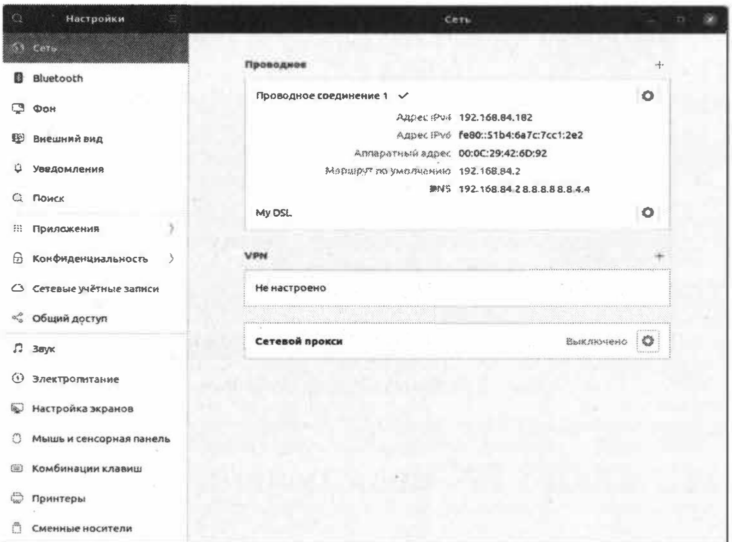


Рис. 7.1. Настройки, Сеть

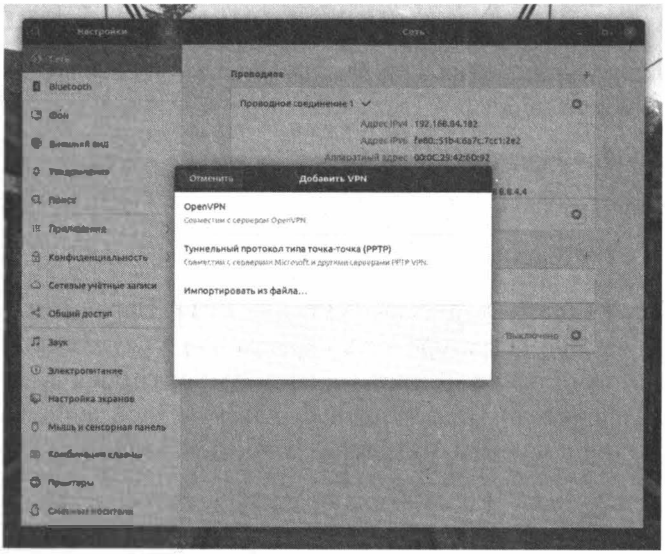


Рис. 7.2. Выбор типа протокола

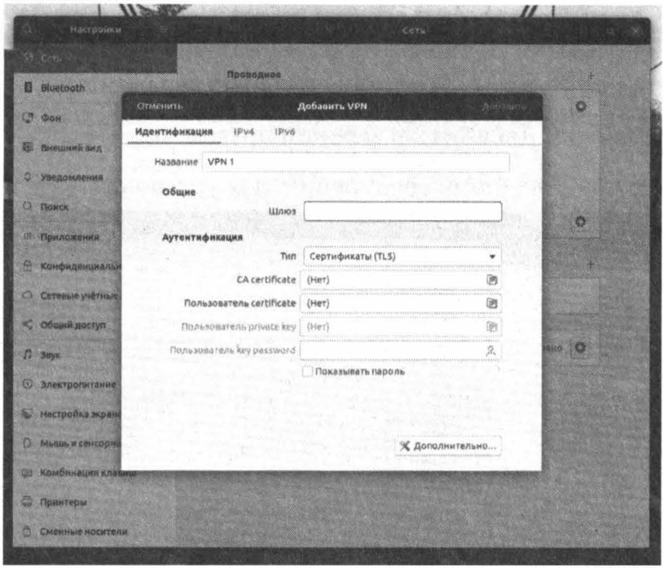


Рис. 7.3. Параметры соединения

7.3. Настройка VPN-подключения в Astra Linux

Настроить VPN-соединение можно так:

1. Из меню **Network Manager** выберите команду **Соединения VPN, Добавить VPN-соединение** (рис. 7.4)
2. Появится окно, в котором нужно или выбрать пункт **OpenVPN** для ручной настройки или же выбрать **Импортировать сохраненную конфигурацию VPN** для использования сгенерированного при настройке учетной записи файла
3. При ручной настройке вам нужно указать IP-адрес или имя VPN-сервера, указать сертификаты и пароль пользователя (рис. 7.6). Нажав кнопку **Дополнительно**, есть возможность указать дополнительные параметры, например, порт соединения, если используется нестандартный
4. Нажмите кнопку **Сохранить** для создания соединения. Оно появится в меню Network Manager и его можно будет выбрать для установки соединения

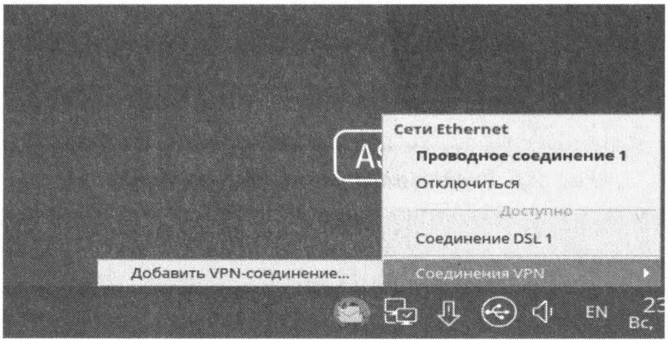


Рис. 7.4. Меню Network Manager

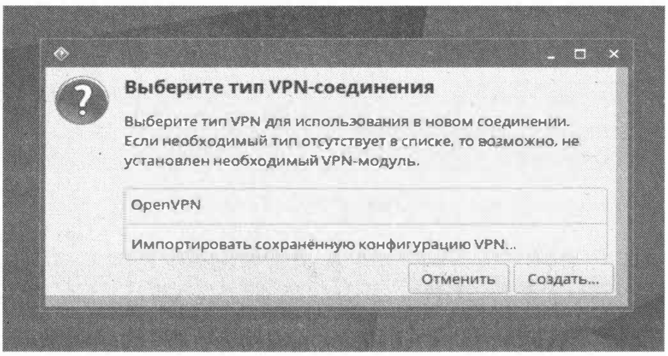


Рис. 7.5. Выбор типа соединения

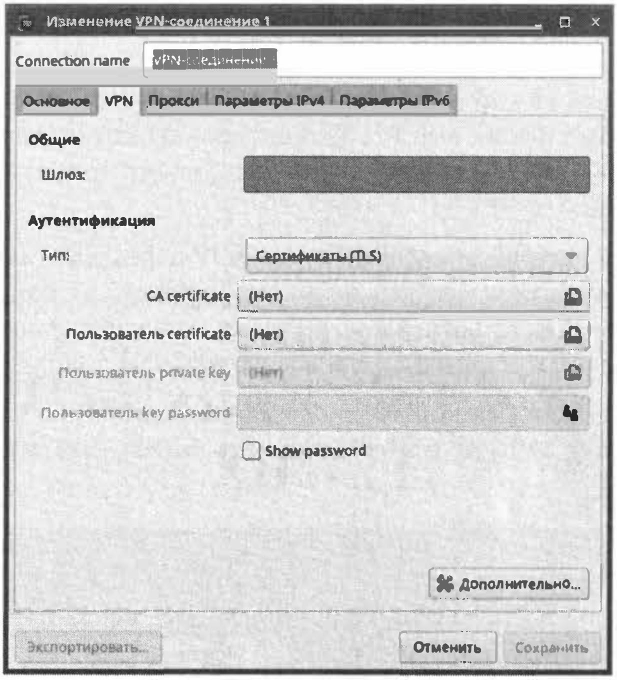


Рис. 7.6. Ручная настройка VPN-соединения

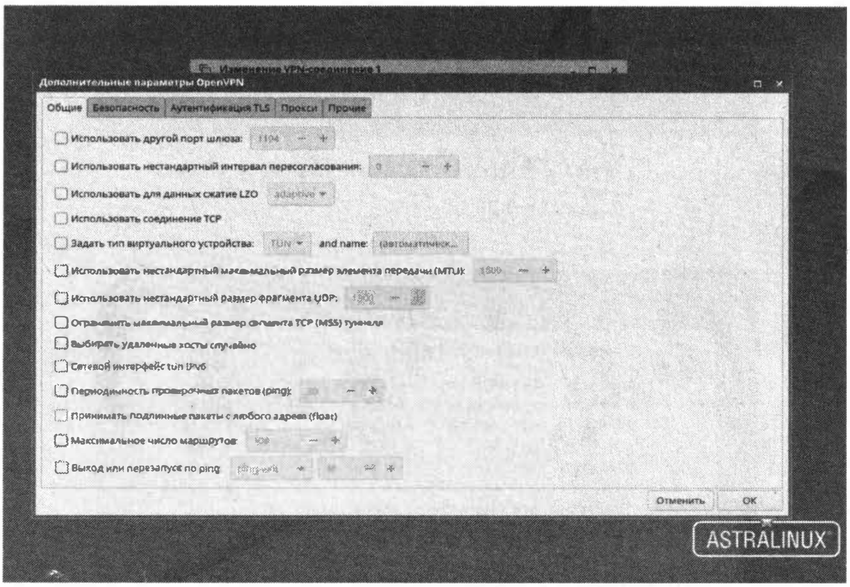
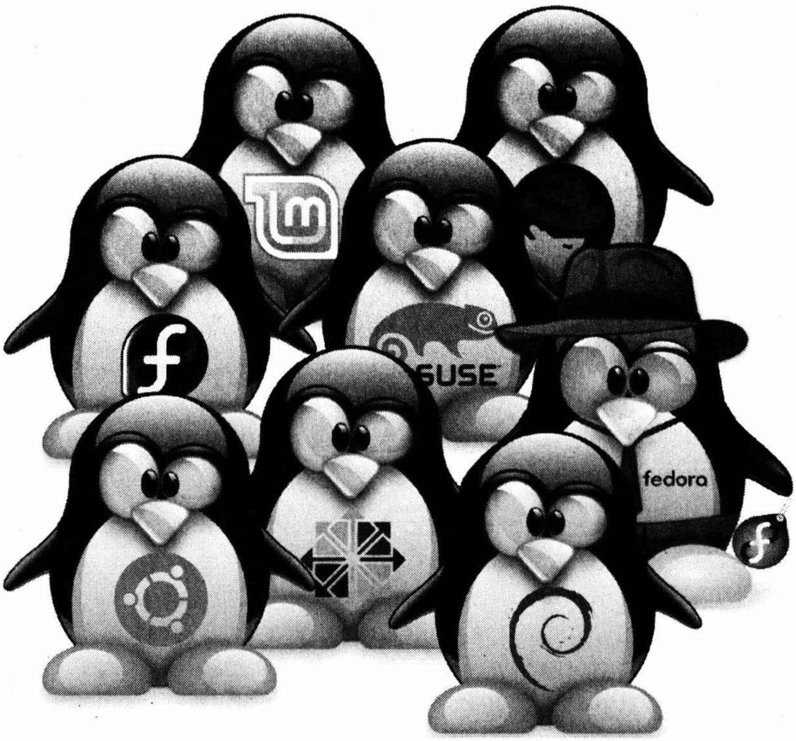


Рис. 7.7. Дополнительные параметры VPN-соединения

Глава 8.

Установка программ в Linux



8.1. Способы установки программ

Существует *три способа* установки программ: компиляция из исходных кодов, из пакетов, из снапов. Первый способ устарел и сейчас используется очень редко – когда нет другого способа установить ту или иную программу на свой компьютер. Выглядит все это так: вы сначала устанавливаете программное обеспечение разработчика (заголовочные файлы, компилятор **gcc**, библиотеки, различные утилиты вроде **make** и т.д.), затем загружаете и распаковываете архив с исходными кодами программы. Запускаете компиляцию и ждете, пока программа будет "собрана" из исходников. Даже если все пройдет гладко и вам не придется адаптировать исходные коды (что требует определенных знаний) под свою систему, процесс компиляции занимает много времени. Удалить "установленную" таким образом программу можно только вручную.

Сегодня все мыслимое и немыслимое программное обеспечение поставляется в уже откомпилированном виде. *Самостоятельная сборка (компиляция)* программного обеспечения имеет смысл только в случаях, если вам нужно установить программу, когда для вашей архитектуры[1]/дистрибутива нет пакета или же когда вы хотите получить самую последнюю версию программы – когда разработчики еще не успели создать пакет для вашего дистрибутива.

1 Хотя есть большая вероятность того, что раз для вашей архитектуры нет пакета, то у вас не получится откомпилировать программу

Установка из пакета гораздо проще. Вы получаете пакет с программой и устанавливаете его. Пакет содержит уже откомпилированную программу и необходимые для работы этой программы файлы, например, файлы конфигурации, страницы руководства и др. При удалении пакета вся эта информация будет централизовано удалена и вам не придется исследовать каждый каталог системы и вручную удалять остатки программы.

Максимум, что может пойти не так при установке пакета – это нарушение зависимостей. Об этом мы поговорим далее в этой главе. Забегая наперед, обычно для удовлетворения зависимостей нужно удалить или установить какие-то другие пакеты. Это несложно и в некоторых случаях система сама справляется с поставленной задачей. Установка программы из пакета занимает считанные секунды, в крайнем случае – минуты, большая часть времени уходит на загрузку самого пакета из Интернета.

Получить пакет можно, как вручную, собственноручно скачав его из сайта разработчика, так и воспользоваться менеджером установки пакетов, который сам скачает пакет из репозитария и установит его.

Третий способ, снапы – относительно недавний. Как уже было отмечено, при установке пакета могут понадобиться дополнительные пакеты. Как правило, это пакеты с какими-то библиотеками. Иногда происходит так, что нужная библиотека отсутствует в вашем дистрибутиве как таковая или конфликтует с имеющейся версией библиотеки. Особенно часто такое происходит, если вы пытаетесь установить пакет не из репозитария своего дистрибутива, а скачанный с сайта разрабтчика. Чтобы избавиться от подобных танцев с бубном, были придуманы снапы. Снап содержит свою программу и все необходимые для ее работы библиотеки. При этом программа, установленная со снапа, будет использовать не системную версию библиотеки, а ту версию, которая шла со снапом. В результате программа будет выполняться корректно и ее установка никак не отразится на других программах, поскольку процесс установки никак не затрагивает имеющиеся в системе библиотеки. О снапах мы еще поговорим отдельно в этой главе.

8.2. Типы пакетов и их содержимое

Существует два формата пакетов – DEB (расширение .deb) и RPM (.rpm). Первые используются в Debian-совместимых дистрибутивах – Ubuntu, Astra Linux, Mint, ALT Linux и др. Второй тип используется в RedHat-совместимых – RHEL, Fedora, CentOS, openSUSE и т.д.

Независимо от формата пакетов в самом пакете кроме устанавливаемой программы и вспомогательных файлов содержится различная служебная информация: информация о разработчике, о версии программного продукта, информация о зависимостях и конфликтах, пути для установки (указывают, куда должны быть скопированы файлы, имеющиеся в пакете в процессе установки пакета).

Отдельного разговора заслуживает информация о зависимостях и конфликтах. Некоторые пакеты для своей работы требуют установки дополнительных пакетов. Например, пакет ubuntu-desktop требует для своей работы множества других пакетов – alsa-base (файлы звукового драйвера), alsa-utils, bc, anacron и т.д. Говорят, что пакет ubuntu-desktop зависит от пакетов alsa-base, alsa-utils и т.д. В свою очередь эти пакеты могут зависеть от других пакетов. При установке пакета **ubuntu-desktop** менеджер пакетов выполнит разрешение зависимости, то есть установит все пакеты, от которых зависит наш пакет и все пакеты, от которого зависят другие устанавливаемые пакеты. Может получиться, что для установки одной небольшой программы будет установлено очень много других пакетов, от которых зависит эта программа. Но ничего не поделаешь – если программа вам нужна, то придется пойти на это. Как вариант – найти аналогичную программу.

Некоторые программы могут находиться в системе только в единственном экземпляре. Например, почтовый агент может быть установлен только один, иначе между ними произойдет конфликт. Поэтому пакет А может конфликтовать с пакетом Б. Если у вас установлен А, а вы пытаетесь установить Б, менеджер пакетов сообщит о конфликте и предложит или отказаться от установки или перед установкой пакета Б удалить пакет А.

8.3. Источники пакетов

Список источников пакетов в порядке убывания их популярности:

- **Репозитарии** – каждый дистрибутив работает с собственным репозитарием (хранилищем) пакетов. Да, для установки программного обеспечения вам понадобится доступ к Интернету, но это самый современный способ установки. Репозитарии позволяют эффективно управлять обновлением программного обеспечения. Используя репозитарии, вы можете быть уверенными, что у вас будет установлено самое новое программное обеспечение.

- **Установочный диск** – ранее дистрибутивы Linux распространялись на DVD-дисках (некоторые – на одном, некоторые – на нескольких). На DVD, кроме самой системы, была и львиная доля программного обеспечения, которая только может понадобиться пользователю. Все программное обеспечение при установке системы не устанавливалась, но была возможность установки с инсталляционного носителя после установки. Когда доступ к Интернету был медленным и дорогим, пользователи предпочитали устанавливать пакеты с DVD-диска. Преимущество этого способа в том, что не нужно загружать пакеты с удаленного сервера, что экономило время и деньги. Сейчас же доступ к Интернету в большинстве случаев высокоскоростной (даже 10 Мбит/с считается высокоскоростным) и безлимитный, поэтому чаще всего на DVD-диске содержится только самое необходимое программное обеспечение, а все остальное загружается из Интернета. К тому же, ПО с DVD наверняка устареет к моменту его установки, а из репозитария будет загружена более новая версия – пакеты в репозитариях периодически обновляются.
- **Сайт разработчика программного обеспечения** – иногда программное обеспечение не включено в состав репозитария дистрибутива. Это может произойти по разным причинам. Например, программное обеспечение является проприетарным (коммерческим) или же просто узкоспециализированным и его не стали включать в состав дистрибутива. В этом случае вам нужно самостоятельно скачать пакеты и установить их. В некоторых случаях разработчики предоставляют доступ к собственным репозитариям, из которых можно загрузить все необходимые пакеты.

8.4. Менеджеры пакетов

Изначально в RedHat и Debian использовались программы **rpm** и **dpkg**. Сейчас обе эти программы все еще входят в состав дистрибутивов, но использовать их крайне не рекомендуется. Дело в том, что эти программы ничего не "знают" о зависимостях между пакетами. Если для работы пакета А нужен пакет Б, то программа даже толком не сообщит о том, какой пакет нужен для установки пакета А. Она сообщит о том, что для установки пакета А нужна, например, библиотека lib40, а в каком DEB-пакете находится эта библиотека и откуда ее нужно заполучить – вам придется догадаться самому.

Также программы **rpm** и **dpkg** ничего не знают о репозитариях, поэтому удел этого менеджера – установка пакета из локального источника. Желательно, чтобы для работы устанавливаемого пакета не требовались другие пакеты –

ведь о зависимостях эти программы, как уже было отмечено, ничего не знают. В репозитариях содержится информация обо всем дереве зависимостей. Например, пакет А может зависеть от пакета Б, пакет Б – от пакетов В, Г и Д. Менеджеры пакетов, такие как **apt** установят все необходимые пакеты в необходимом порядке, чего не дождешься от программ **rpm** и **dpkg**.

Именно поэтому для управления пакетами рекомендуется использовать менеджеры пакетов. Вот неполный список возможностей таких программ:

- Поиск пакетов в репозитариях;
- Установка пакетов из репозитариев;
- Установка/удаление пакетов с разрешением зависимостей и конфликтов;
- Обновление пакетов.

Принцип работы менеджера пакетов простой. Представим, что вы устанавливаете пакет А. Менеджер производит список вашего пакета по списку всех установленных репозитариев. В одном из репозитариев ваш пакет точно будет, в противном случае менеджер сообщит о том, что пакет не найден. Далее менеджер смотрит на список зависимостей и если есть пакеты, от которых зависит наш пакет А, то они также загружаются и устанавливаются – до установки пакета А. Затем производится загрузка и установка самого пакета А.

Установка программного обеспечения в Linux требует прав *root*. Поэтому нужно сначала запускать любые команды управления программным обеспечением через команду *sudo*. Например:

```
sudo apt install apache2
```

В дистрибутивах Debian, Ubuntu, Astra Linux и других, которые основаны на Debian, используется менеджер пакетов **apt**. В других дистрибутивах могут использоваться другие менеджеры, например, в Fedora используется **dnf**, в CentOS и старых версиях Fedora – **yum**. В openSUSE используется свой менеджер пакетов **zypper**.

Далее будет рассмотрен менеджер пакетов **apt**. При установке пакетов **apt** разрешает зависимости. Вы просто указываете, какой пакет вы хотите установить, а что будет происходить дальше – уже не ваша забота. Вас интересует конечный результат. Лишь бы требуемый пакет был в репозитариях.

> **Примечание.** В старых версиях Debian и Ubuntu использовалась команда apt-get вместо apt. Основные параметры (см. табл. 8.1) у этих двух команд такие же, нет смысла рассматривать устаревшую версию менеджера пакетов. Если вам нужно работать со старой версией, обратитесь к странице руководства (man apt-get).

Список репозитариев хранится в файле /etc/apt/sources.list. Содержимое этого файла показано на рис. 8.1.

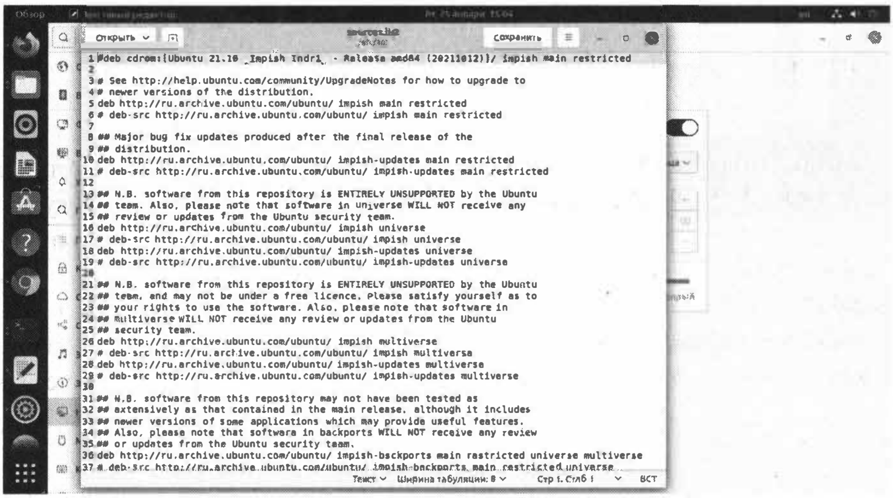


Рис. 8.1. Файл sources.list

Вы вряд ли будете вручную редактировать этот файл. Исключения могут составить лишь ситуации, когда вы добавляете сторонний репозитарий, содержащий какую-то программу, чтобы установить ее впоследствии с помощью **apt**. Что же касается стандартных репозитариев, то ними проще управлять (включать/выключать) посредством приложения **Программы и обновления**, изображенном на рис. 8.2. На вкладке **Программное обеспечение Ubuntu** вы можете включить/выключить стандартные репозитарии. Также можно включить использование в качестве источника пакетов инсталляционный DVD-диск Ubuntu. Обратите внимание: вы можете не только включить/выключить репозитарии, но и выбрать местоположение

сервера, с которого будут пакеты загружаться. На рис. 8.2 показано, что пакеты будут загружаться с сервера, который находится в РФ.

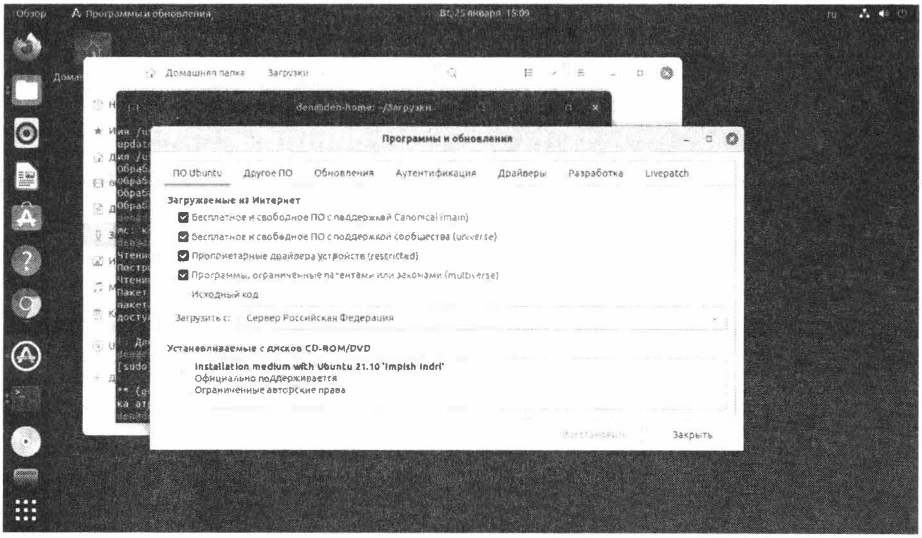


Рис. 8.2. Программы и обновления

Формат вызова команды **apt** следующий:

```
sudo apt [опции] команды [пакет]
```

В качестве примера приведу команду установки пакета **mc**:

```
sudo apt install synaptic
```

Обратите внимание, что я запускаю команду от имени *root*. Если запустить команду `apt` через команду *sudo* (*sudo apt install synaptic*), нужно, чтобы пользователь, который запускает *sudo*, был внесен в файл /etc/sudoers.

По умолчанию пользователь, которого вы создаете при установке Debian, не вносится в этот файл, поэтому нужно или получать права *root* командой *su*, или добавить пользователя в файл /etc/sudoers.

Основные команды `apt` приведены в таблице 8.1.

Таблица 8.1. Основные команды менеджера пакетов apt-get

Команда	Описание
install <список пакетов>	Устанавливает пакеты из списка. Элементы списка разделяются пробелами
remove <список пакетов>	Удаляет пакеты из списка. Элементы списка разделяются пробелами
purge <список пакетов>	Удаляет не только пакеты, но их конфигурационные файлы. Это означает, что если вы установили какую-нибудь программу, настроили ее, а потом удалили командой apt remove, то конфигурационный файл этой программы останется в системе. Если вы теперь установите эту программу снова, то можно будет использовать предыдущий конфигурационный файл, так как он не был удален
check	Поиск нарушенных зависимостей
clean	Очищает локальное хранилище полученных пакетов. При установке пакеты из репозитария загружаются в каталог var/cache/apt/archive. При интенсивной установке программного обеспечения в этом каталоге накапливается довольно много пакетов, поэтому очистка хранилища помогает сэкономить дисковое пространство
upgrade [список пакетов]	Обновляет указанные пакеты, если пакеты не заданы, обновляет все пакеты, требующие обновления
full-upgrade	Обновляет всю систему
update	Синхронизирует внутреннюю базу данных о пакетах с источниками пакетов, которые описаны в /etc/apt/sources.list

autoremove	Когда вы устанавливаете пакет, то часто устанавливаются дополнительные пакеты, являющиеся его зависимостями. Если теперь вы удалите этот пакет, то зависимости останутся в системе. Команда *apt autoremove* удаляет эти зависимости, но только те, которые не нужны другим установленным пакетам
list	Выводит список пакетов, соответствующих какому-то критерию. Примеры приведены далее. Вывести список установленных в системе пакетов: `apt list --installed` Вывести список пакетов, которые требуют обновления (у которых вышла новая версия): `apt list --upgradable` Вывести список всех пакетов доступных для вашей системы: `apt list --all-versions`
show <пакет>	Выводит информацию о пакете
search <слово>	Данная команда выполняет поиск указанного слова в названии пакетов и в описании пакетов. Поддерживаются регулярные выражения
edit-sources	Открывает файл /etc/apt/sources.list в текстовом редакторе для редактирования, после сохранения изменений и закрытия редактора, выполняет проверку файла на предмет ошибок. В случае наличия ошибок, выводит предложение на повторное редактирование файла, чтобы исправить ошибки

8.5. Графические средства установки программ

Установка программ из пакетов имеет свои недостатки. Самый главный из них – вам нужно знать имя пакета, в котором находится нужная вам программа. Если вы следуете какому-то руководству, то особых проблем нет – открыл терминал, ввел команду установки нужного пакета и все. Когда вы знаете имя пакета или команду установки, все хорошо. Но если таких познаний у вас нет, тогда вам нужно или найти имя пакета в Интернете или же воспользоваться графическими средствами для установки программ.

В состав дистрибутива Astra Linux входит, на наш взгляд, один из самых удобных графических менеджеров пакетов – Synaptic. Вы найдете его в программной группе **Системные**. Использовать Synaptic очень просто (рис. 8.3). Слева отображаются группы пакетов и фильтры (под группами пакетов), позволяющие отфильтровать список пакетов по разделам, состоянию, архитектуре и т.д. В правой верхней части находится список пакетов в выделенной группе. Если выделить любой из пакетов, в область ниже будет загружено его описание.

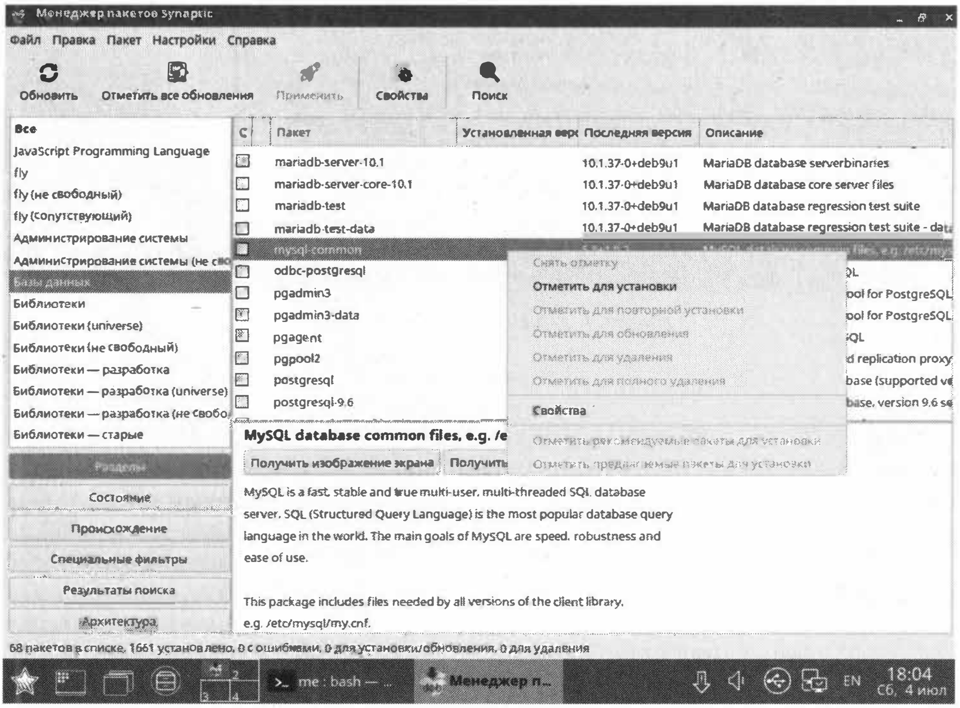


Рис. 8.3. Менеджер пакетов Synaptic

Установленные пакеты отмечаются зеленым квадратиком. У неустановленных квадратик неокрашенный. Чтобы установить пакет, щелкните на нем правой кнопкой мыши и выберите команду **Отметить для установки** (рис. 8.3). Значок пакета будет изменен на квадратик с желтой стрелкой. Можете выбрать другие пакеты, которые вы хотите установить. Когда будете готовы, нажмите кнопку **Применить** для установки пакетов (рис. 8.4).

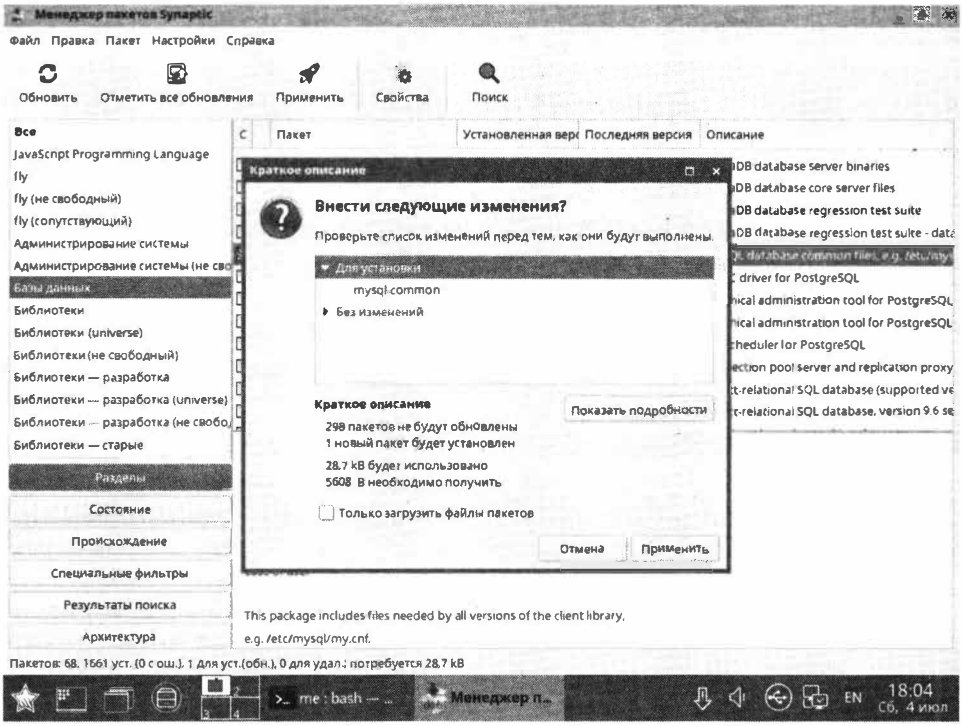


Рис. 8.4. Применение изменений

Менеджер пакетов покажет, какой размер будет загружен из Интернета и сколько пакетов будет установлено. Дождитесь установки пакета (рис. 8.5). После этого вы можете использовать установленную программу.

В Ubuntu менеджер Synaptic по умолчанию недоступен. Для его установки нужно ввести команду:

```
sudo apt install synaptic
```

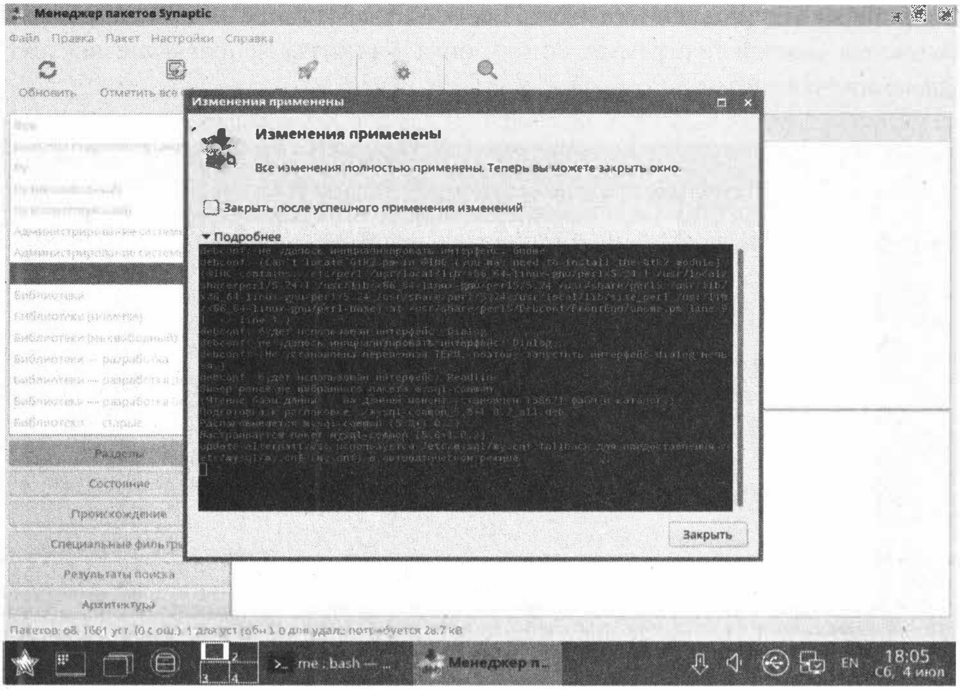


Рис. 8.5. Пакет установлен

Однако в Ubuntu обычному пользователю больше понравится использовать Ubuntu Software или центр программного обеспечения. Все доступное ПО разбито на группы, вы можете выбрать программу и установить ее.

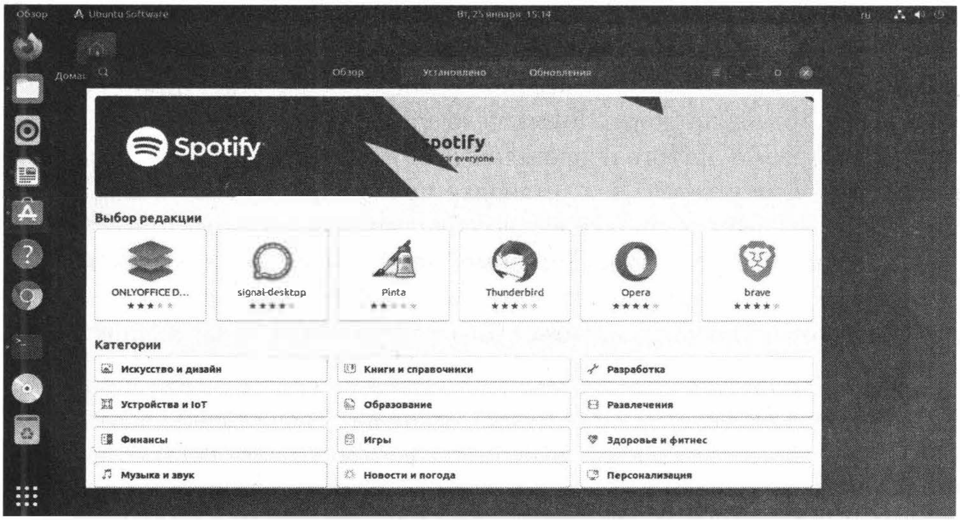


Рис. 8.6. Ubuntu Software

На вкладке **Установлено** находится список уже установленных программ. Напротив каждой программы есть кнопка **Удалить**, использующаяся для удаления программы.

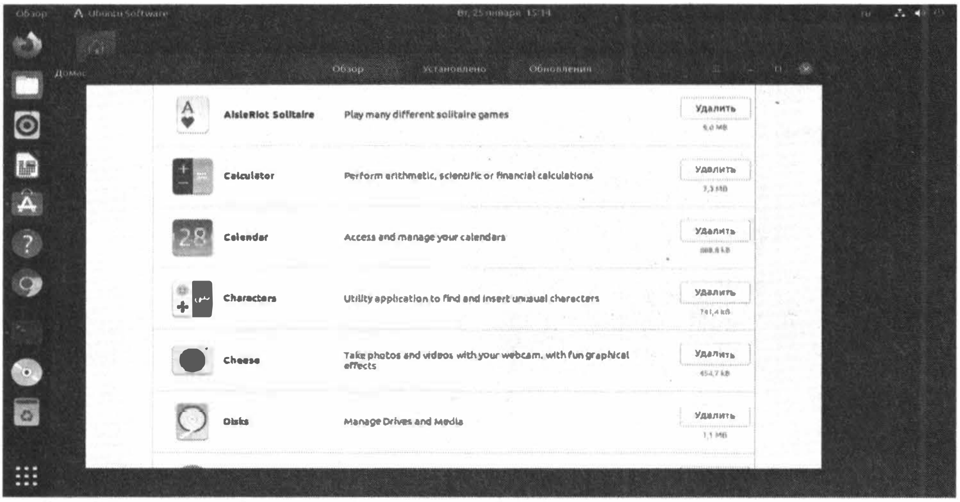


Рис. 8.7. Установленные программы

8.6. Снапы

Ранее уже было сказано, зачем используют *снапы* (snaps). Представим вполне реальную ситуацию. Пользователь устанавливает самую новую версию Ubuntu, пусть это будет версия 20.04. После установки пользователь не обновляет систему. Его все устраивает, и он спокойно работает некоторое время, скажем, год или полтора. Потом он хочет установить новую версию браузера или какого-то другого приложения и не может этого сделать, поскольку его дистрибутив устарел. Для установки приложения нужны новые версии библиотек, а для их установки нужно обновить уже установленные пакеты. Иногда процесс настолько масштабный, что приходится обновлять дистрибутив. А ведь мы знаем, что пользователь не хочет этого делать по ряду причин, да и это опасно: система может быть разрушена из-за нарушения связей между программами и библиотеками.

Есть и другая ситуация – когда нужно установить приложение, пакет которого конфликтует с уже установленным пакетом. Просто устанавливаемое приложение использует библиотеки, которые не могут "ужиться" с теми, которые уже установлены.

Проблемы настолько частые, то были предложены снапы. Пакет содержит саму программу, а также различные вспомогательные файлы – документацию, ресурсы (картинки, например), файлы локализации, какие-то сценарии и т.д. Но пакет не содержит всего, что нужно для работы этой программы в системе. Например, если программе для работы нужна библиотека **lib**, то просто в пакете "прописывается" зависимость – нужно установить такой-то пакет для работы этого пакета. При установке программы менеджер пакетов (apt) производит разрешение зависимостей – устанавливает все необходимые для работы этой программы пакеты. С одной стороны, такой подход позволяет экономить место на диске. Ведь одну и ту же библиотеку не нужно устанавливать несколько раз. С другой стороны, это порождает уже описанные ранее проблемы.

Снап – это решение всей головной боли, как пользователя, так и разработчика приложения. Снап можно считать таким пакетом, в котором содержится не только программа, но и все необходимые для ее работы библиотеки. Получается, что все, что нужно для работы программы содержится в снапе. Да, это неэкономно, но, согласитесь, жестким диском размером в 1 Тб сегодня никого не удивишь, а головной болит будет меньше. Вы просто установите программу и будете ее использовать, а не два дня пытаться решить проблемы, возникшие при установке ее пакета.

Также не нужно бояться, что система превратится в мусорку. Снапы устанавливаются в отдельной папке в виде защищенного от записи образа. Изменения, которые должен внести снап в файловую систему (например, в каталог /lib) будут внесены в виртуальной файловой системе. Таким образом, установка снапа никак не повлияет на работу основной системы и на другие снапы. Снапы решают проблемы с совместимостью версий разных приложений.

Для начала работы со снапами в старых версиях Ubuntu (например, в 16.04, в более ранних версиях они не поддерживаются вообще) нужно сначала установить пакет **snapd**. В современных версиях данный пакет уже установлен.

Для поиска доступных снапов введите команду:

```
snap find <название>
```

Например

```
snap find hello-world
```

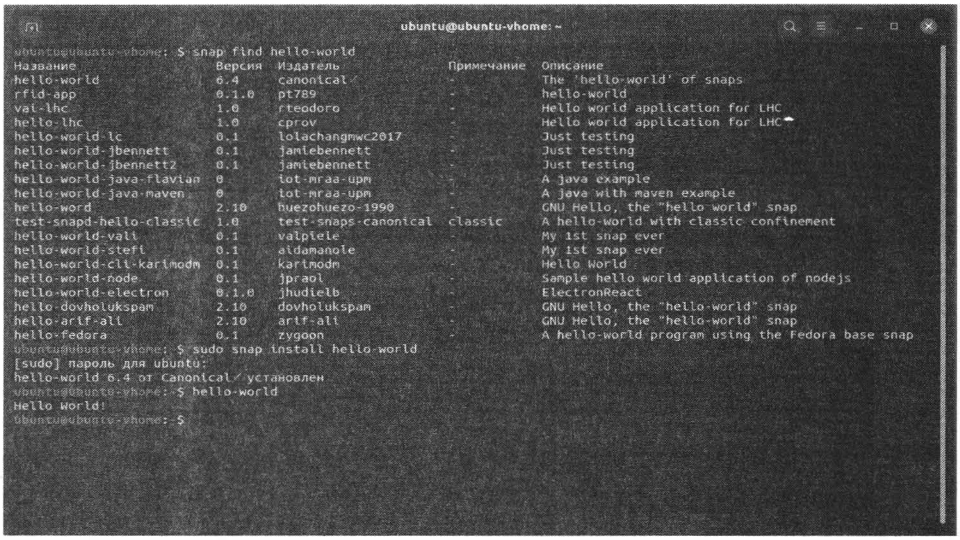


Рис. 8.8. Поиск, установка и запуск снапа

Вывод этой команды изображен на рис. 8.8. В первой колонке приводится название снапа. Пусть мы хотим установить снап *hello-world*:

```
sudo snap install hello-world
```

Как видите, команда установки снапа аналогична команде установки пакета, только вместо *apt* используется утилита *snap*.

После установки снапа можно запустить имеющуюся в нем программу:

```
hello-world
Hello, world!
```

Просмотреть установленные снапы можно командой *snap list* (рис. 8.9).

Как и пакеты, снапы можно обновить. Например:

```
sudo snap refresh hello-world
```

Данная команда обновит снап *hello-world*, если для него доступны обновления. Это очень удобно – вы обновляете не только приложения, но и все необходимые для его работы библиотеки.

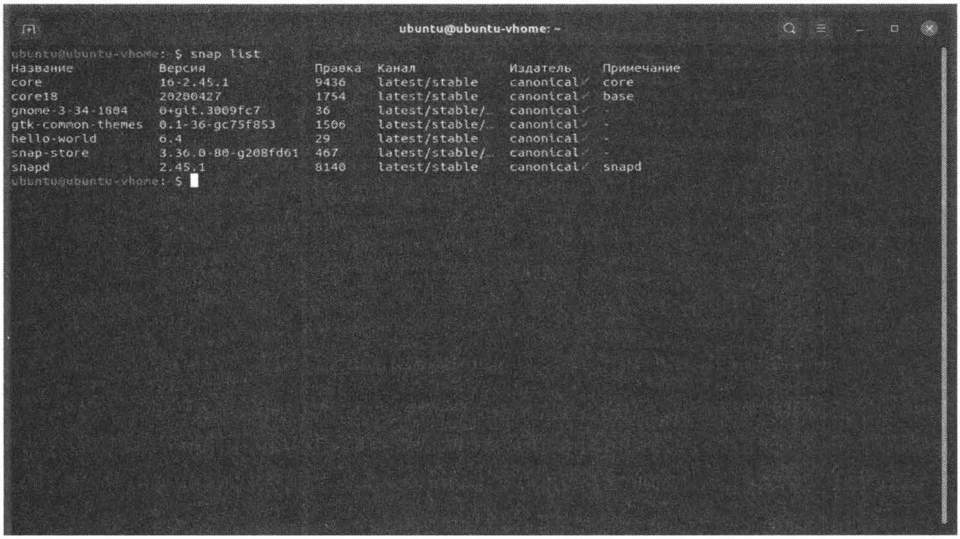


Рис. 8.9. Команда snap list

Для обновления всех снапов используется другая команда:

```
sudo snap resresh
```

Теперь вы знаете, для чего используются снапы и что это вообще такое. А вот обновлять систему или использовать снапы – каждый решает сам.

8.7. Ошибка при выполнении apt: *Unable to acquire the dpkg lock /var/lib/dpkg/lock*

К сожалению, время от времени при работе с Linux возникают различные ошибки. Одна из наиболее часто возникающих звучит так: **Не удалось получить доступ к файлу блокировки /var/lib/dpkg/lock-frontend - open (11: Ресурс временно недоступен)**. Ошибка возникает при попытке установить пакет в Ubuntu с помощью команды apt install.

Сообщение об ошибке может немного отличаться в зависимости от различных условий. Например, могут появляться следующие ошибки:

```
E: Could not get lock /var/lib/dpkg/lock - open (11: Resource temporarily
unavailable)
E: Unable to lock the administration directory (/var/lib/dpkg/), is another
process using it?
E: Could not get lock /var/lib/apt/lists/lock - open (11: Resource temporarily
unavailable)
E: Unable to lock directory /var/lib/apt/lists/
E: Could not get lock /var/lib/dpkg/lock - open (11: Resource temporarily
unavailable)
E: Unable to lock the administration directory (/var/lib/dpkg/), is another
process using it?
```

Данные ошибки появляются, когда программа **apt** не может получить доступ к файлу блокировки /var/lib/dpkg/lock*. Данный файл используется, чтобы запретить одновременное выполнение операций, связанных с управлением пакетами в системе, так как при одновременном изменении данных о пакетах будет нарушена целостность "пакетной базы".

Обычно существует две основные причины появления, описанных ошибок:

1. В данный момент уже выполняется экземпляр программы **apt**.
2. Предыдущий вызов **apt** завершился некорректно.

Сначала нужно проверить, что уже не запущен другой экземпляр программы apt-get (apt). Выполним следующую команду, чтобы проверить есть ли **apt** в списке запущенных процессов:

```
ps aux | grep -i apt
```

Вывод может быть таким:

```
ubuntu 8425 0.0 0.0 79516 3752 pts/1 S+ 10:31 0:00 sudo aptt install inkscape
ubuntu 8456 0.0 0.0 38892 944 pts/0 S+ 10:32 0:00 grep --color=auto -i apt
```

В первой строке мы видим, что уже есть работающий экземпляр программы **apt**, который имеет PID (идентификатор) 8425. Вторая строка относится к нашей команде *grep*, которую мы запустили с аргументом *apt*, поэтому она вывела саму себя. Итак, нас интересует только первая строка.

Если вы уверены, что не запускали программу **apt** сами, или она не запущена в фоновом режиме, например, выполняется автоматическое обновление системы, то нужно принудительно завершить ее выполнение. Для этого воспользуемся командой *kill −9*. Команде нужно указать числовой идентификатор процесса. В нашем случае это 8425:

```
sudo kill -9 8425
```

После выполнения данной команды, процесс с идентификатором 8425 завершится.

Если первый способ вам не помог, то нужно удалить все файлы блокировки. Для этого выполняем команды:

```
sudo rm /var/lib/apt/lists/lock
sudo rm /var/cache/apt/archives/lock
sudo rm /var/lib/dpkg/lock
sudo rm /var/lib/dpkg/lock-frontend
```

Если при выполнении каких-нибудь из этих команд появится сообщение: `rm: невозможно удалить '/var/../lock': Нет такого файла или каталога`, это нормально, не обращайте на него внимания.

После этого нужно выполнить переконфигурацию пакетов:

```
sudo dpkg --configure -a
```

8.8. Невозможно найти определенный пакет

При работе над книгой мы столкнулись со следующей ситуацией. Когда вы вводите команду установки пакета, например, ту же `sudo apt install synaptic`, то менеджер пакетов сообщает вам, что такой пакет не найден. Проблема оказалась в том, что список пакетов почему-то не синхронизировался с сервером ru.ubuntu.com. Для ее решения запустите *update-manager*, нажмите кнопку **Настройки** и настройте менеджер пакетов на использование основного сервера, как показано на рис. 8.10. Не факт, что такая проблема проявится у вас, но если так произойдет, то вы знаете, что с этим делать.

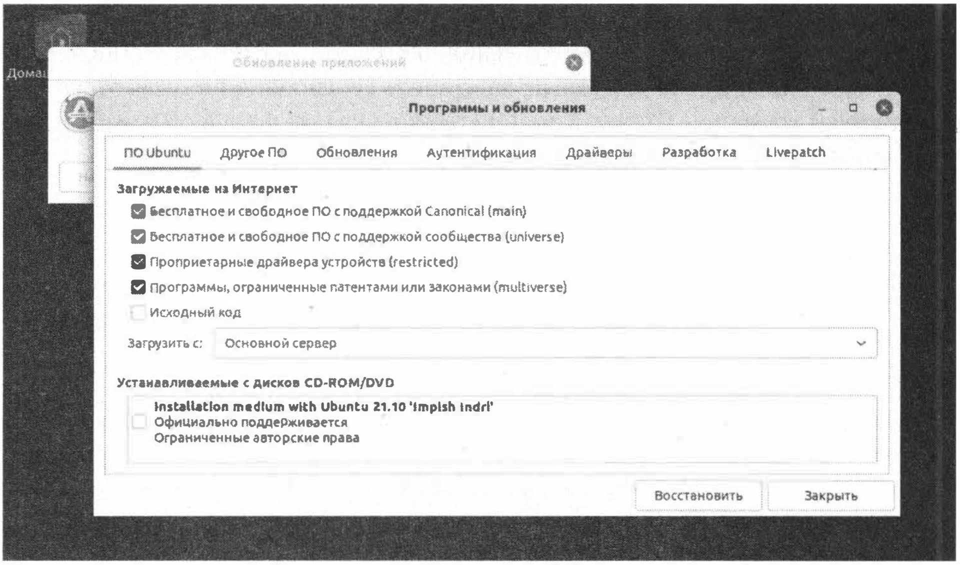
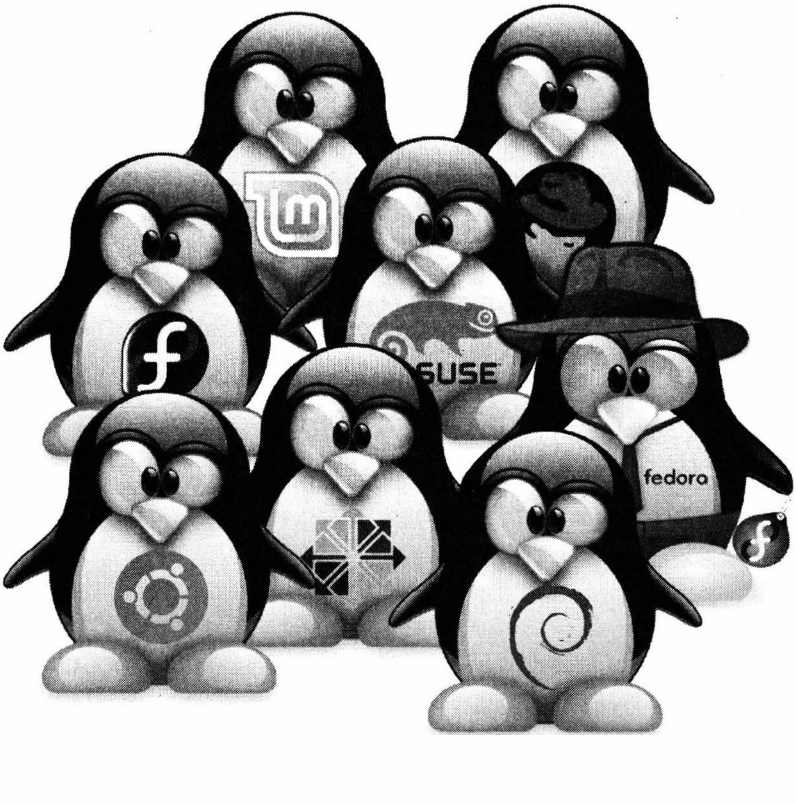


Рис. 8.10. Использование основного сервера пакетов

После этого введите команду *sudo apt update* для обновления списка пакетов. Как только список пакетов будет обновлен, повторите попытку установки пакета.

Глава 9.

Популярные программы



9.1. Офисные пакеты

Когда-то в Linux было многообразие офисных пакетов. Со временем все эти пакеты были вытеснены одним – LibreOffice, в котором содержатся все необходимые аналоги программ из MS Office:

- LibreOffice Writer – текстовый процессор, аналог MS Word;
- LibreOffice Calc – электронная таблица, аналог MS Excel;
- LibreOffice Impress – презентации, аналог MS PowerPoint;
- LibreOffice Draw – векторный графический редактор для создания блок-схем и диаграмм, аналог MS Visio;
- LibreOffice Base – база данных, аналог MS Access.

Данные программы не просто выполняют те же функции, что и программы из пакета MS Office, но и поддерживают форматы MS Office. Да, интерфейс программ несколько специфичен (напоминает старые версии MS Office), но при желании можно разобраться. К тому же есть версия LibreOffice для Windows. Офисный пакет полностью бесплатный, поэтому есть неплохая возможность сэкономить, если установить его еще и на Windows-машины.

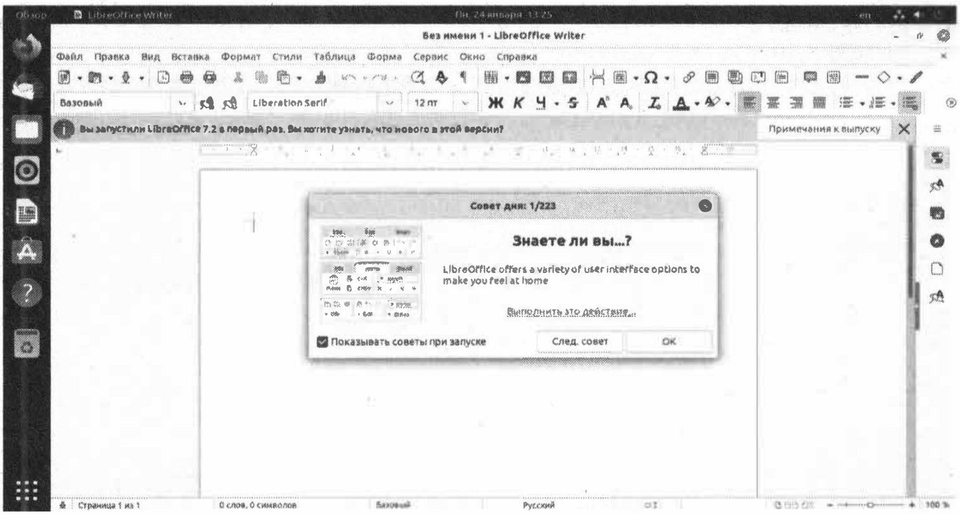


Рис. 9.1. Текстовый процессор LibreOffice Writer

Если же какого-то функционала MS Office будет не хватать, тогда не забывайте об онлайн-версии Office 365, которую можно использовать прямо в браузере.

9.2. Графические текстовые редакторы

Для редактирования обычного текста (без форматирования), например, сценариев командной оболочки или конфигурационных файлов, можно обойтись стандартным текстовым редактором. Для редактирования файлов конфигурации нужны права root, поэтому запускать редактор нужно из терминала так:

```
sudo gedit <имя_файла>
```

Штатный текстовый редактор обладает возможностью поиска и замены текста, нумерует строки, показывает позицию курсора. Для редактирования текстовых файлов конфигурации – то, что нужно.

Если возможностей этого редактора вам окажется мало, используя Ubuntu Software, вы всегда сможете установить редактор Atom (рис. 9.3). В Ubuntu Software он называется **snappyatom**. Это профессиональный текстовый

редактор для программистов и благодаря системе плагинов, его возможности практически безграничны (рис. 9.4).

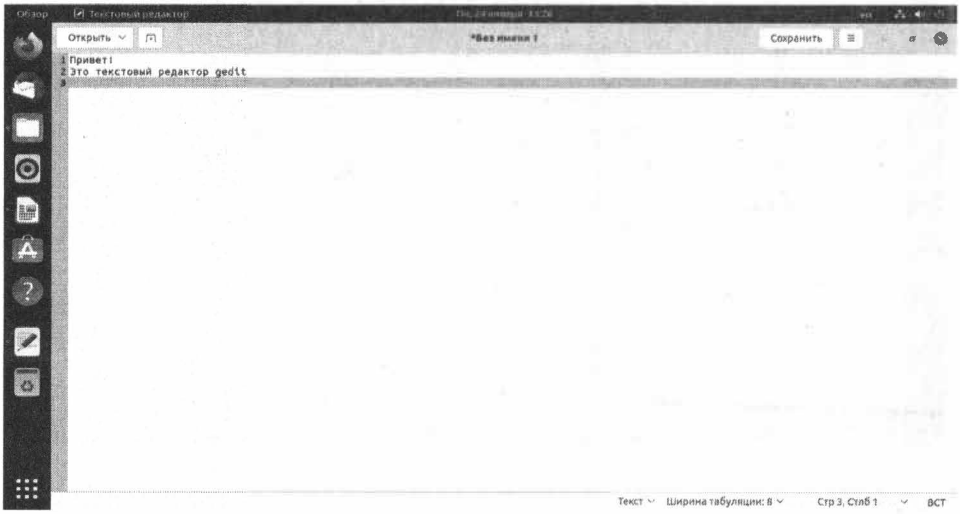


Рис. 9.2. Текстовый редактор gedit

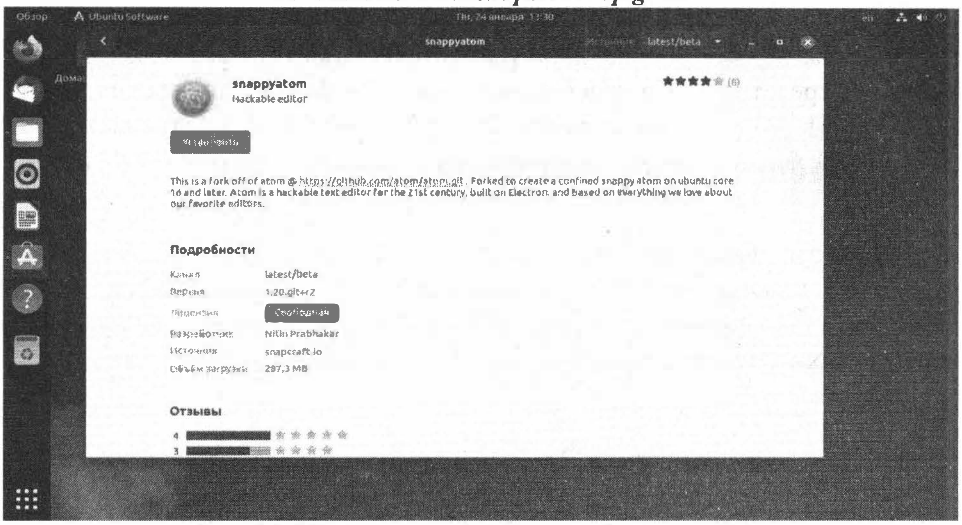


Рис. 9.3. Установка текстового редактора Atom

9.3. Консольные текстовые редакторы

Работая в графическом режиме, вам вряд ли захочется использовать консольные текстовые редакторы. Но при администрировании виртуального сервера выбора у вас не будет.



Рис. 9.4. Текстовый редактор Atom

Виртуальные серверы Linux не оснащены каким-либо графическим интерфейсом, поэтому их администрирование осуществляется через консоль – или посредством Web-консоли, встроенной в панель управления, или же по SSH. Часто у администратора возникает потребность отредактировать какой-то файл конфигурации сервера. Проблем с этим, как правило, никаких нет – запускаешь предпочитаемый текстовый редактор, открываешь файл, редактируешь и сохраняешь. Однако редактирование некоторых файлов конфигурации, в частности /etc/sudoers, осуществляется только посредством специальных утилит (в данном случае – visudo или crontab – при редактировании расписания планировщика), которые запускают текстовый редактор по умолчанию. Таковым редактором является редактор **vi**, перекочевавший в современные дистрибутивы Linux с 1970-ых годов и его нельзя назвать удобным. В этой заметке будут рассмотрены некоторые текстовые редакторы, и будет показано, как по умолчанию установить понравившийся редактор, чтобы он вызывался при редактировании некоторых специальных файлов конфигурации, которые нельзя редактировать вручную.

Самый удобный — редактор **nano** (раньше он назывался **pico** и входил в состав почтового клиента **pine**). Редактор **nano** изображен на рис. 9.5.

Внизу (под текстом) есть подсказка по комбинациям клавиш для управления редактором. Символ ^ означает <Ctrl>. То есть для выхода из редактора нужно нажать комбинацию клавиш <Ctrl>+<X>, а для сохранения текста — <Ctrl>+<O>.

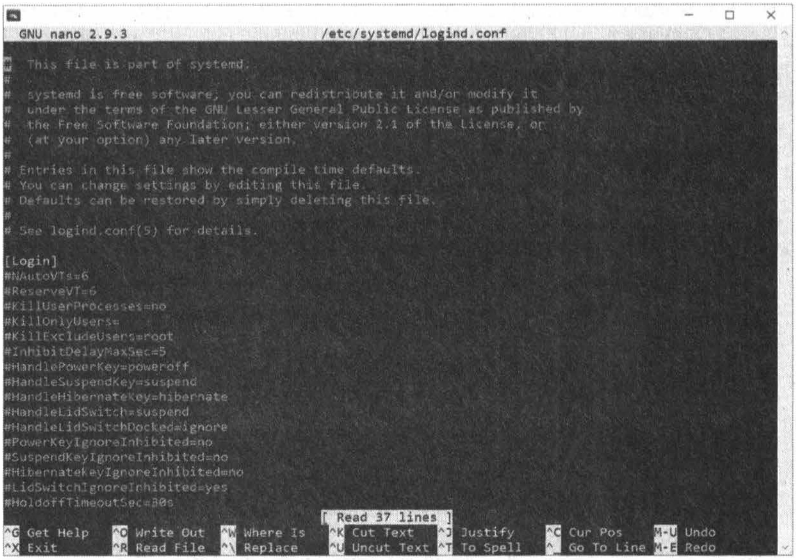


Рис. 9.5. Текстовый редактор nano

В некоторых системах (например, в FreeBSD) вместо **nano** используется редактор **ee** (в Linux его нет). Он похож на **nano**, но подсказки выводятся до текста (вверху экрана), а не после него, но идея та же. Также довольно удобен редактор **joe**. Скажем так, **nano** будет удобнее, он поддерживает подсветку синтаксиса, внизу есть панель с подсказками, но это дело привычки.

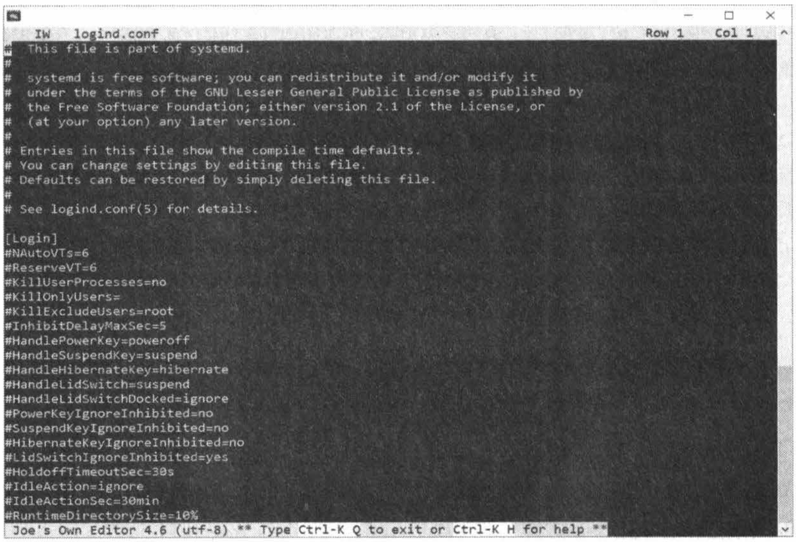


Рис. 9.6. Редактор joe

В пакет **mc** (файловый менеджер) входит довольно удобный редактор **mcedit**, который запускается при нажатии клавиши **<F4>** в **mc** (рис. 9.7). При желании вы можете запустить редактор отдельно:

```
mcedit <имя файла>
```

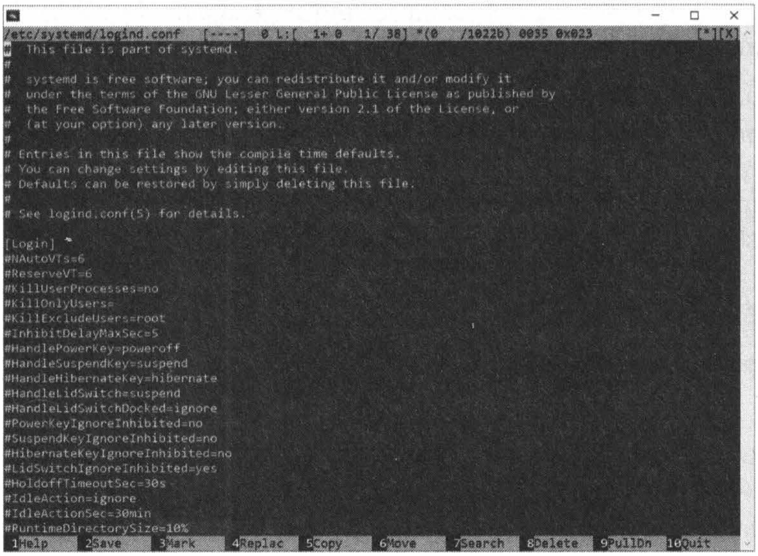


Рис. 9.7. Редактор mcedit

Классический синий фон, подсказки функциональных клавиш внизу и т.д. Редактор не менее удобен, чем **nano**.

Кстати, редакторы **joe**, **nano** и **ee** запускаются аналогично:

```
joe <имя файла>
nano <имя файла>
ee <имя файла>
```

Выбор конкретного редактора зависит от ваших личных предпочтений. Но в любом случае, каждый из представленных редакторов будет удобнее, чем стандартный **vi**.

Некоторые утилиты, например, *crontab*, visudo вызывают текстовый редактор по умолчанию для редактирования тех или иных данных. В этом случае будет вызван **vi**, который, как было отмечено, неудобен. Чтобы вызывался

нужный вам редактор, его нужно сделать редактором по умолчанию. Для этого нужно установить переменную окружения EDITOR:

```
which nano
/bin/nano
export EDITOR=/bin/nano
```

Первая команда (*which nano*) сообщает путь к выбранному редактору. Далее этот путь нам нужно указать в качестве значения переменной EDITOR.

Вот только помните, что при следующем входе в систему переменная EDITOR будет установлена по умолчанию. Чтобы этого не произошло, нужно отредактировать файл .bashrc того пользователя, от имени которого будете редактировать конфигурационные файлы. В случае с root это будет файл /root/.bashrc:

```
cd ~
nano .bashrc
```

В этот файл нужно добавить команду:

```
export EDITOR=/bin/nano
```

Сохраните файл. Теперь нужно выйти из системы (команда *exit*) и снова войти (по SSH или через Web-консоль). После этого запустите любую команду, вызывающую стандартный текстовый редактор, например, crontab –e. Если вы увидели выбранный вами текстовый редактор, значит, все прошло нормально (рис. 9.8). В противном случае вы где-то допустили ошибку.

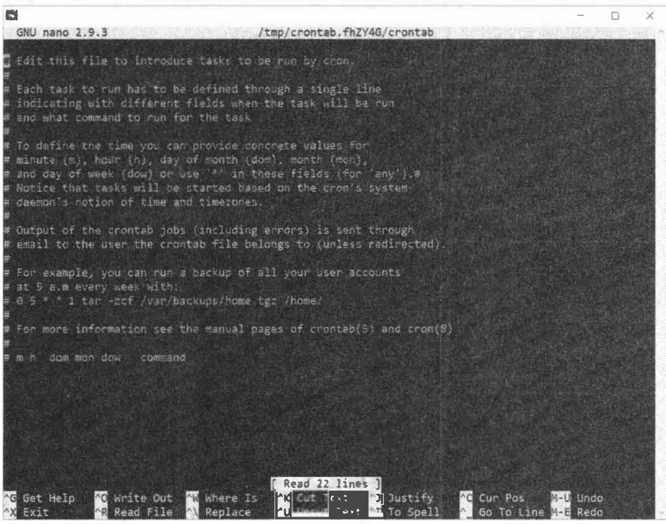


Рис. 9.8. При редактировании расписания (команда crontab –e) открылся nano. Настройка успешна

9.4. Программы для работы с Интернетом

Основные программы для работы с Интернетом установлены по умолчанию. К ним относятся:

- Браузер Firefox;
- Почтовый клиент Thunderbird;
- Torrent-клиент Transmission.

При желании вы можете доустановить sFTP Client (клиент обмена файлами по протоколу SSH). К преимуществам sFTP Client можно отнести наличие терминала SSH, что пригодится, если вы надумаете удаленно администрировать Linux-сервер. Установка этой программы осуществляется посредством **Ubuntu Software** и предельно проста – все, что вам нужно сделать – нажать кнопку **Установить**.

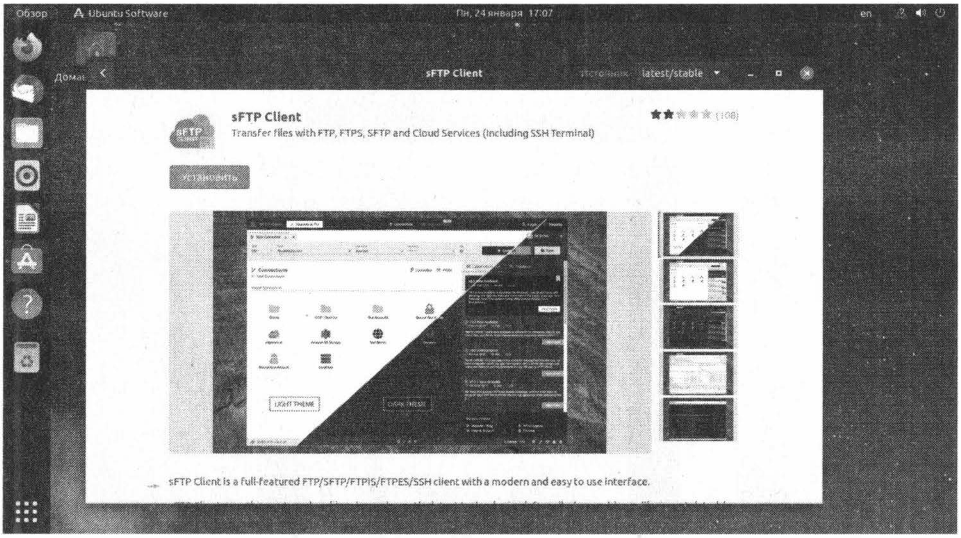


Рис. 9.9. Ubuntu Software

Ранее было показано, как установить популярный браузер Chrome, если вам не понравится Firefox.

Также вам, скорее всего, захочется установить Skype. Сделать это также можно посредством Ubuntu Software (рис. 9.10). Программа устанавливается просто и без всяких танцев с бубном, как это было в предыдущих версиях Ubuntu. Сразу после установки Skype готов к использованию (рис. 9.11).

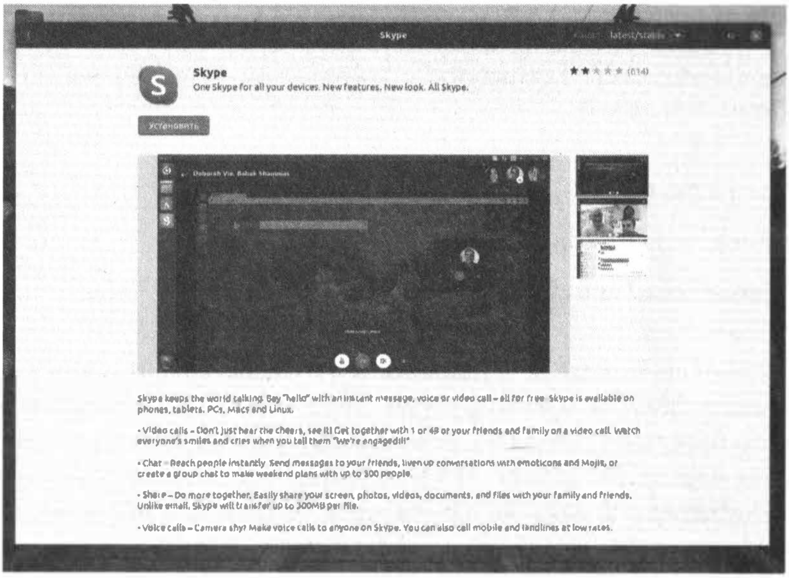


Рис. 9.10. Установка Skype



Рис. 9.11. Skype запущен

Популярный мессенджер Viber также доступен для Linux и также устанавливается посредством Ubuntu Software (рис. 9.12).

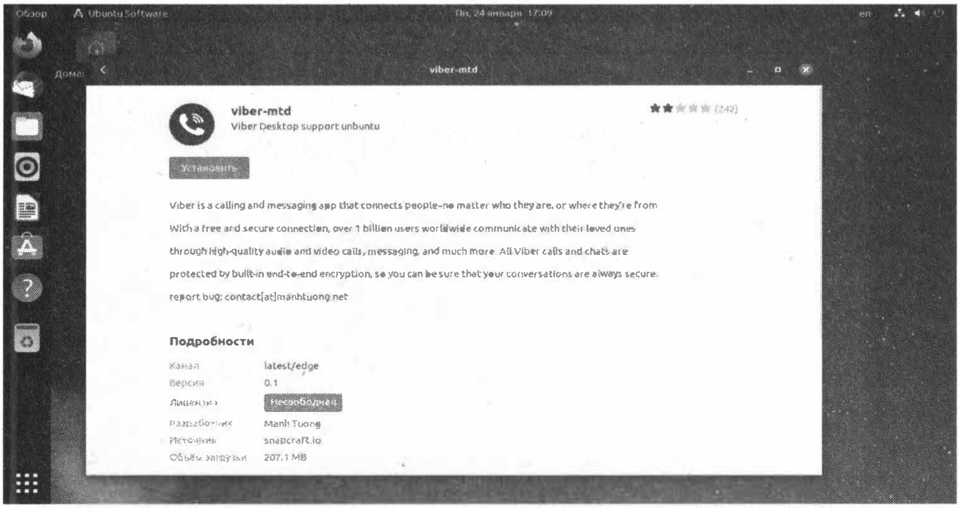


Рис. 9.12. Установка Viber

Заметьте, вы только что установили много программного обеспечения и не ввели ни одной команды. В предыдущих версиях Ubuntu вам пришлось бы ввести с десяток команд, не говоря уже о том, чтобы заставить установленные программы работать. На данный момент у вас есть большинство программ, которые могут понадобиться при работе в Интернете обычному пользователю.

9.5. Linux-аналоги Windows-программ

Новичкам в Linux не всегда просто, поскольку они попросту не знают, какое приложение установить. Не всегда есть Linux-версии известных программ, но в большинстве случаев есть несколько аналогов. Таблица 9.1 поможет вам выбрать Linux-программу на все случаи жизни.

Таблица 9.1. Аналоги Windows-программ

Название Windows-приложения	Linux-аналоги
ACDSee	gThumb F-Spot Shotwel DigiKam PhotoQt GPicView

Acronis True Image	CloneZilla
Adobe Audition	Ardour
Adobe Illustrator	LibreOffice Fraw Inkscape sK1 Xara Xtreme
Adobe Photoshop	GIMP Krita Photivo
Adobe Lightroom	Darktable
Adobe Premiere Pro	Open Movie Editor PiTiVi
Adobe Reader	Xpdf Evince Google Chrome
Alcohol 120	Furius ISO Mount
AnyDesk	Remmina
Apple iCloud for Windows	iCloud for Linux
Ardour	Ardour
Audacity	Ardour
Avira Antivirus	ClamAV Avira AntiVir Personal Avast Antivirus Linux Home Edition
Blender	Blender
CINEMA 4D	Blender

Corel Draw	Inkscape LibreOffice Draw sK1 Xara Xtreme
CPU Z	I-Nex CPU-X CPU-G
CuteFTP	FOFF FileZilla gFTP
CyberLink PowerDVD	Totem
Daemon Tools	Furius ISO Mount Etcher
Darktable	Darktable
DaVinci Resolve	Avidemux
Deluge	Transmission
Discord	Discord
Dr.Web Antivirus	Avira AntiVir Personal Avast Antivirus Linux Home Edition ClamAV
Dropbox	Dropbox
eMule	aMule
ESET NOD32 Antivirus	Avira AntiVir Personal Avast Antivirus Linux Home Edition ClamAV
Etcher	Etcher

Evernote	Boostnote MyNotex Zim CherryTree
FileZilla	FileZilla
FineReader	gImageReader
Foxit Reader	Xpdf Evince Aesop
Freehand	sK1
GeoGebra	GeoGebra
GOG	Steam
Google Chrome	Google Chrome
Google Earth	Google Earth Marble
InterVideo WinDVD	MPlayer Totem
IrfanView	gThumb GPicView Mirage Ristretto
LMMS	Ardour
Maple, Magma, Mathematica, Matlab	Sage
Maya	Blender

Microsoft Access	LibreOffice Base
Microsoft Excel	LibreOffice Calc IBM Lotus Symphony
Microsoft Office Publisher	LibreOffice Draw
Microsoft OneNote	CherryTree
Microsoft Outlook	Evolution Thunderbird Sylpheed
Microsoft Outlook Express	Evolution Thunderbird Sylpheed
Microsoft Paint	Gnome Paint Pinta
Microsoft PowerPoint	IBM Lotus Symphony LibreOffice Impress
Microsoft Visio	LibreOffice Draw
	Code Blocks KDevelop Visual Studio Code NetBeans Eclipse Qt Creator
Microsoft Windows Journal	Xournal

Microsoft Windows Media Player	MPlayer Lollypop Celluloid Totem GMusicBrowser Clementine Rhythmbox Audacious VLC
Microsoft Windows Movie Maker	Open Movie Editor PiTiVi Flowblade Lightworks Avidemux
Mp3tag	EasyTag
MPV	Celluloid
Nero	K3b Brasero GnomeBaker
Norton Commander	emelFM2 Double Commander GNOME Commander Midnight Commander
Notepad++	JuffEd Medit Atom

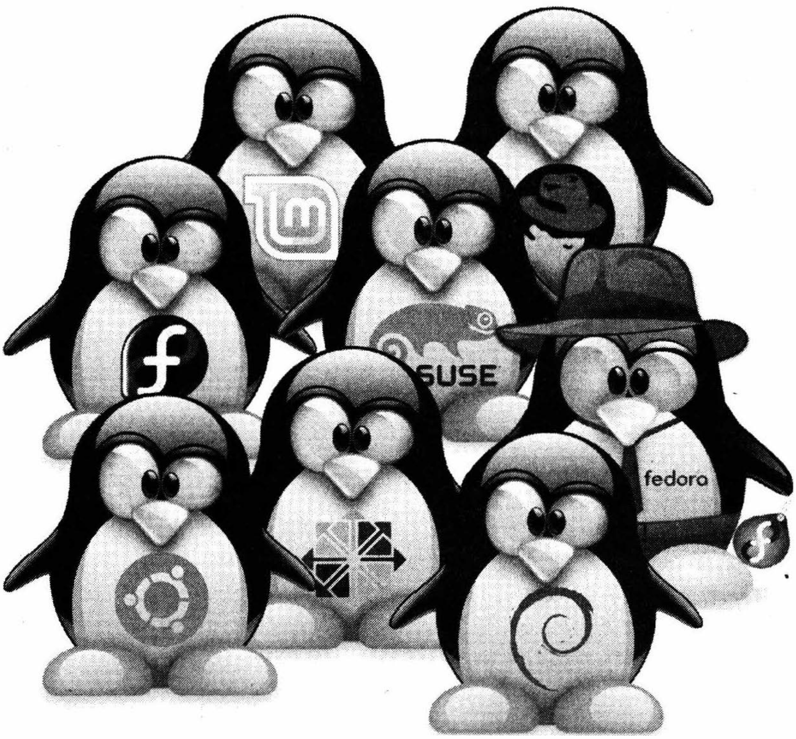
Opera	Opera Google Chrome Epiphany
Partition Magic	GParted
Picasa	DigiKam
qBittorrent	Transmission
Qt Creator	KDevelop
QuarkXPress	Scribus
ReGet	Steadyflow JDownloader
Remote Desktop Connection	Remmina
Rufus	Etcher
Scite	Medit Screem Scribes Bluefish Editor SciTE Komodo Edit Geany
SketchUp	Blender
Skype	Skype
Slack	Discord
SmartFTP	Filezilla
SolidWorks	FreeCAD

Steam	Steam
TeamViewer	Remmina
The Bat	Thunderbird Claws Mail
Total Commander	4Pane GNOME Commander Midnight Commander Krusader emelFM2
UltraISO	Etcher
Universal USB Installer	Etcher
uTorrent	qBittorrent Deluge Tixati Transmission
Viber	Viber
VirtualDub	PiTiVi Avidemux Open Movie Editor
Visual Studio Code	Visual Studio Code
VLC	VLC
Win32 Disk Imager	Etcher

WinAMP	Decibel Audio Player QMMP Clementine Totem Audacious AlsaPlayer Sayonara MPlayer
Wireshark for Windows	Wireshark
Xara	Inkscape Xara Xtreme
XnView	DigiKam Mirage GPicView PhotoQt gThumb
Zoom	Discord
3D Studio Max	Blender

Глава 10.

Запуск Windows-приложений в Linux



Wine – это программа с открытым исходным кодом, которая позволяет запускать Windows-приложения в среде Linux и MacOS. Можно сказать, что это слой совместимости между операционной системой и Windows-программами. Вызовы процедур из библиотек Windows подменяются на системные вызовы Linux и с помощью этого появляется возможность выполнять Windows-программы в Linux.

Обратите внимание Wine: не виртуальная машина, в которую устанавливается Windows (которую, по-хорошему, нужно лицензировать). Это платформа запуска Windows-приложений, которой не нужна сама Windows! Соответственно и лицензировать ничего не нужно.

Платформа Wine постоянно развивается, постоянно выходят новые версии, в которых больше поддерживаемых функций Windows, исправлены многие ошибки, добавляется поддержка новых возможностей. Стабильные релизы Wine выходят приблизительно раз в год, полтора. Но корректирующие, тестовые релизы выпускаются постоянно, даже по несколько раз в месяц.

За последнее время Wine очень сильно продвинулся в плане запуска игр. Благодаря библиотеке DXVK уже можно играть даже многие современные игры Windows без потери производительности. Дальше будет рассмотрена

установка wine Ubuntu 20.04. Установим последнюю версию из официальных репозитариев, а также воспользуемся PPA.

10.1. Установка из официального репозитария

Для установки из официального репозитария просто введите команду:

```
sudo apt install wine
```

> **Внимание!** При первой установке пакета вы можете столкнуться с ситуацией, что пакет не найден. Для решения проблемы нужно обновить список пакетов. Введите команду *sudo apt update*, а затем повторите попытку установки пакета.

> **Внимание!** Если при установке пакета **apt** вам сообщается, что файл блокировки заблокирован каким-то процессом (при этом сообщается номер этого процесса, например, 1203), то выполните команду kill <номер_процесса> (она "убьет" процесс, вызвавший блокировку) и после этого повторите попытку установки пакета. Обе эти ситуации изображены на рис. 10.1. На рис. 10.2 – нормальная установка пакета.



Рис. 10.1. Нештатные ситуации

Установка займет много времени, поскольку нужно будет загрузить много вспомогательных пакетов (рис. 10.2). Можете выпить чашку кофе или позвонить другу, в общем, займите себя чем-нибудь.



Рис. 10.2. Установка wine

Если вы увидите сообщение, что пакет **wine** не найден, тогда выполните команду:

```
sudo update-manager
```

В появившемся окне вместо сервера в Российской федерации выберите **Основной сервер**. Как показывает практика, ru.ubuntu.com почему-то время от времени сбоит и не всегда удается загрузить с него пакеты. После этого введите команду:

```
sudo apt update
```

После этого повторите попытку установки **wine**.

10.2. Установка из PPA

В репозитариях Ubuntu содержится далеко не самая последняя версия. На момент написания этих строк в репозитарии impish (версия 21.10) содержится версия 5.0.3 (рис. 10.3).

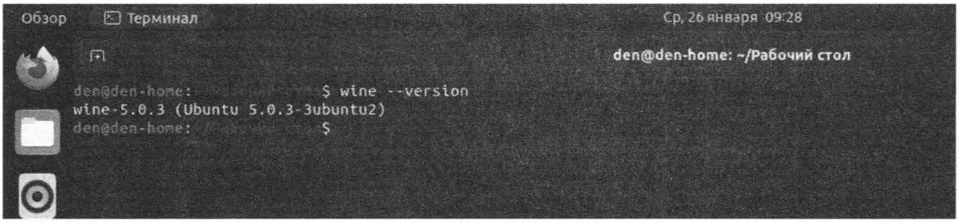


Рис. 10.3. Версия wine, установленная из репозитария

А из репозитария разработчика можно установить версию 6.22 (рис. 10.4) и даже есть версия для разработчиков 7.0. При этом вам не нужно устанавливать тонну ненужного вам программного обеспечения и компилировать программу из исходного кода. Все, что вам нужно – это добавить новый репозитарий и установить из него **wine**.

wine-staging-dev_7.0~rc6~impish-1_amd64.deb	2022-01-14 22:19	3.1M
wine-staging_6.20~impish-1_amd64.deb	2021-10-23 09:08	4.2M
wine-staging_6.22~impish-1_amd64.deb	2021-11-20 20:44	4.2M
wine-staging_6.23~impish-1_amd64.deb	2021-12-04 15:41	4.2M
wine-staging_7.0.0~impish-1_amd64.deb	2022-01-18 19:38	4.3M
wine-staging_7.0~rc1~impish-1_amd64.deb	2021-12-11 05:56	4.3M
wine-staging_7.0~rc2~impish-1_amd64.deb	2021-12-18 06:58	4.3M
wine-staging_7.0~rc3~impish-1_amd64.deb	2021-12-26 19:48	4.3M
wine-staging_7.0~rc4~impish-1_amd64.deb	2022-01-02 20:02	4.3M
wine-staging_7.0~rc5~impish-1_amd64.deb	2022-01-07 21:52	4.3M

Рис. 10.4. Содержимое репозитария

Wine требует поддержку 32-битной архитектуры для установки, поэтому добавим нужную архитектуру командой:

```
$ sudo dpkg --add-architecture i386
```

Загрузим ключ репозитария и добавим его в APT с помощью следующей команды:

```
$ wget -O - https://dl.winehq.org/wine-builds/winehq.key | sudo apt-key add -
```

Добавим репозитарий **impish main** – он содержит пакеты для Ubuntu 20.04, если у вас другая версия, то замените **focal** на название версии Ubuntu:

```
$ sudo add-apt-repository 'deb https://dl.winehq.org/wine-builds/ubuntu/
dists/impish main'
```

Обновим список пакетов:

```
$ sudo apt update
```

Теперь установим **wine**:

```
$ sudo apt install --install-recommends winehq-stable
```

Осталось дождаться установки.

10.3. Настройка после установки

Сразу после установки введите команду:

```
$ winecfg
```

Запустится конфигуратор, основное назначение которого – создать служебные каталоги, в том числе каталог для диска C:, на который будут устанавливаться Windows-программы. В окне конфигуратора выберите версию Windows. По умолчанию используется Windows 7, но учитывая, что современные программы постепенно отказываются от ее поддержки, рекомендуется выбрать Windows 10/11.

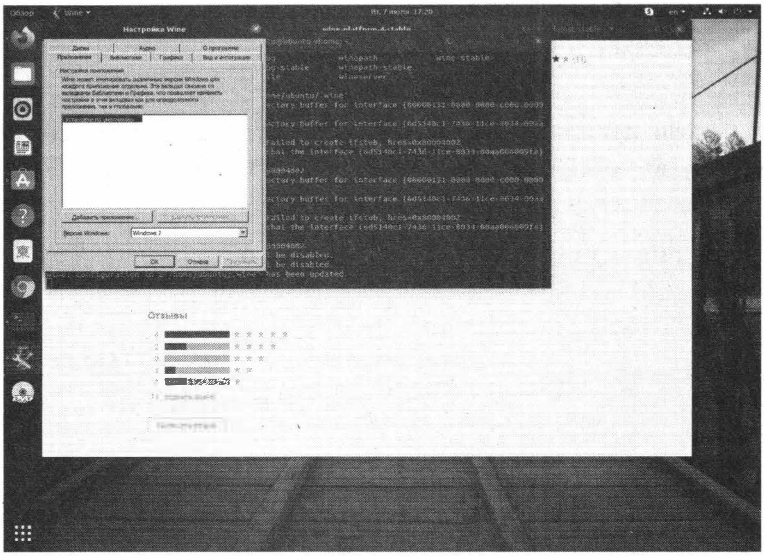


Рис. 10.5. Установка версии Windows

На вкладке **Диски** содержит список дисков, которые будут доступны Windows-приложениями. По умолчанию содержимое диска С: будет находиться в каталоге ~/.wine/drive_c. Остальные параметры можете не изменять – вернетесь к ним, когда у вас появится такая необходимость.

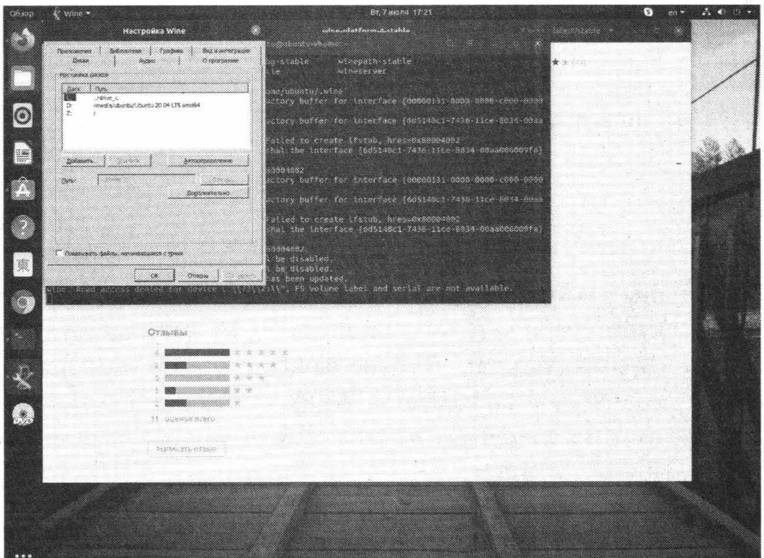


Рис. 10.6. Вкладка Диски

10.4. Установка и запуск Windows-программы

Далее будет показан процесс установки и запуска Windows-программы. Мы будем устанавливать популярных двухпанельный файловый менеджер Total Commander – помимо всего прочего он пригодится вам для просмотра файловой системы, которую "видят" устанавливаемые Windows-приложения.

Скачайте установочный 64-битный exe-файл с официального сайта (https://www.ghisler.com/). Он будет помещен в каталог **Загрузки**. Перейдите в этот каталог, используя файловый менеджер Ubuntu, щелкните правой кнопкой мыши в любой свободной области и выберите команду **Открыть в терминале**. Так вы откроете терминал, в котором уже будет выбран текущий каталог **Загрузки** – так быстрее.

Введите команду:

```
$ wine tcmd951x64
```

Запустится программа установки Total Commander (рис. 10.7). Произведите установку, как обычно. По завершении установки вы получите соответствующее сообщение (рис. 10.8)

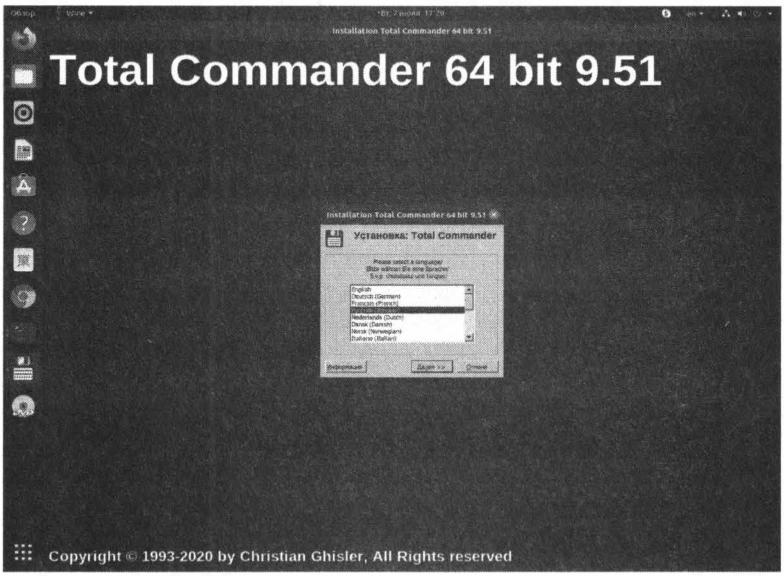


Рис. 10.7. Программа установки Total Commander

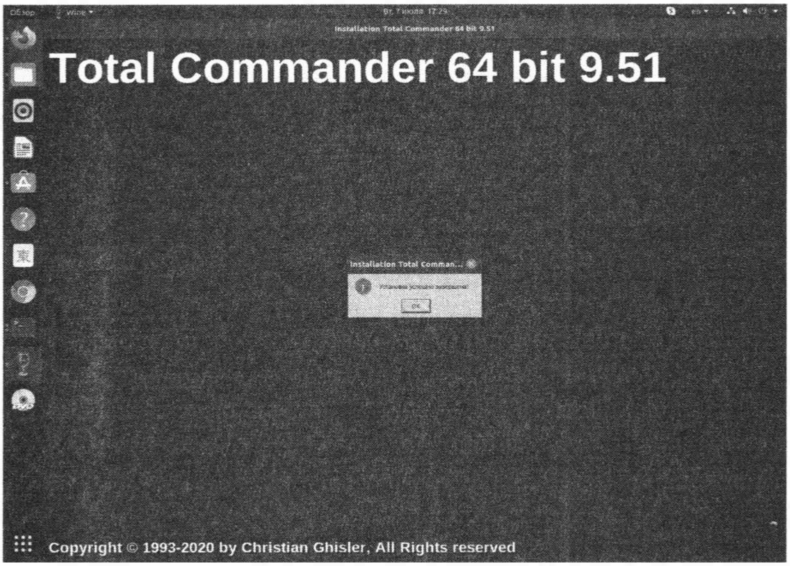


Рис. 10.8. Установка завершена

Далее откройте экран **Приложения**, и вы найдете там кнопку для запуска Total Commander. Прошу заметить: это все-таки Windows-приложение, а запустить вы его можете как обычное "родное" приложение для Linux (рис. 10.9).

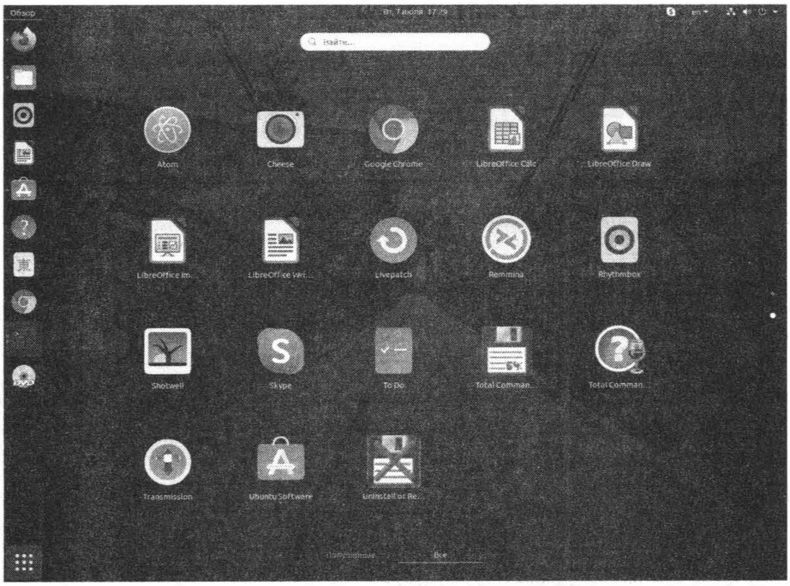


Рис. 10.9. Экран приложения можно использовать для запуска Windows-приложений

Запустите программу (рис. 10.10). Далее вы можете пользоваться ею без всяких ограничений.

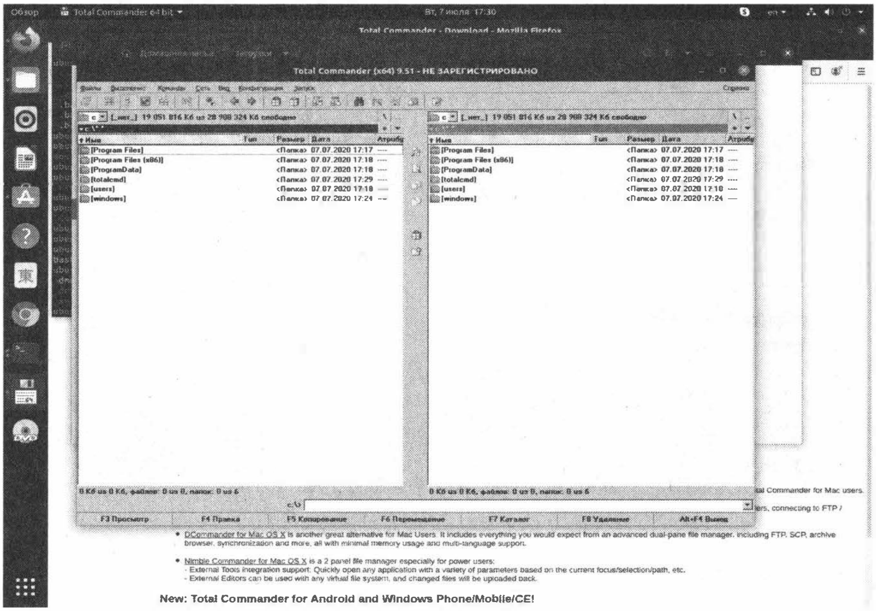


Рис. 10.10. Total Commander запущен в Linux

10.5. Список игр и других приложений, работающих через Wine

Что делать, если не получается запустить приложение в Linux? Во-первых, попробуйте изменить версию Windows. В некоторых случаях это может помочь. Во-вторых, поищите решение на форумах: скорее всего, вы не единственный, кто столкнулся с такой проблемой и наверняка решение уже есть. В-третьих, поищите нужные настройки в базе данных приложений.

На официальном сайте Wine ведется база данных программ и игр, которые можно запустить через Wine: Wine Application Database (AppDB) — https://appdb.winehq.org.

Для каждого приложения можно получить информацию об особенностях установки, запуска и настройки данного приложения через Wine, о проблемах и багах, с которыми можно столкнуться.

База данных постоянно пополняется. В ней насчитывается более 27200 приложений. Вот некоторые из самых популярных программ и игр, работающих через Wine (конкретную версию уточняйте в базе данных AppDB):

- Adobe Animate
- Adobe Photoshop
- Microsoft Office
- Total Commander
- Lingvo
- 1С:Предприятие
- Гарант
- КонсультантПлюс
- Final Fantasy XI Online
- StarCraft
- World of Warcraft
- Warcraft III
- World of Warcraft 8.3.0
- Counter-Strike: Source
- EVE Online
- Half-Life 2
- Magic: The Gathering Online
- The Sims 3
- StarCraft 1.16.1

10.6. Использование отдельных префиксов

Некоторые программы должны запускаться внутри своей среды, то есть должны быть изолированы от других приложений. Для этого им нужен отдельный префикс (отдельная директория среды, в которой они будут работать).

Префикс задается переменной WINEPREFIX. Посмотрим, как все это реализовать на практике. Сначала создадим новый префикс. Выполняем команду:

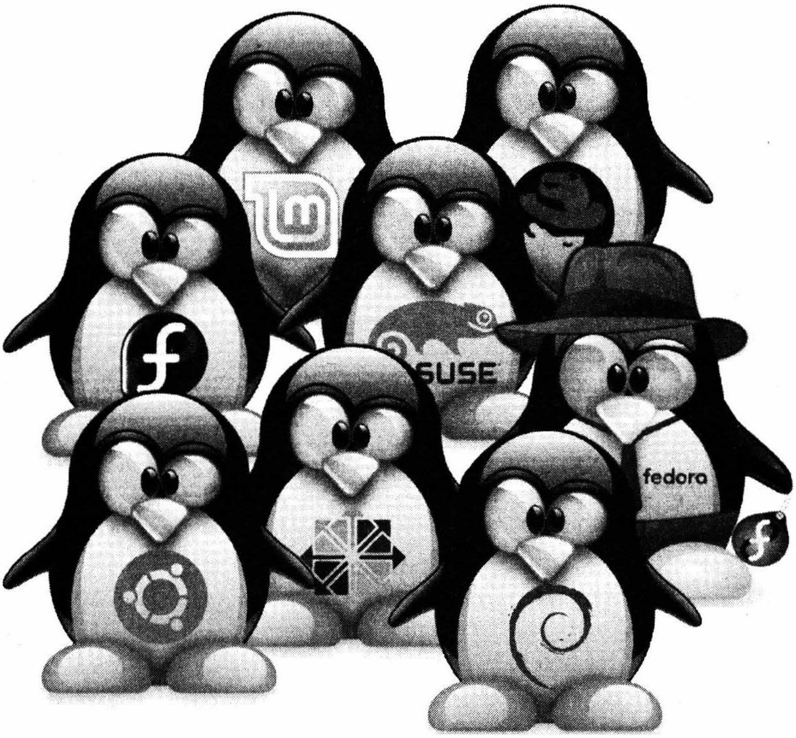
```
WINEPREFIX="/home/ubuntu/.wine/" winecfg
```

Теперь выполняем саму программу и указываем для нее новый префикс:

```
WINEPREFIX="/home/ubuntu/.wine/" wine /путь/к/файлу/setup.exe
```

Глава 11.

Печать документов в Linux



11.1. Добавление и настройка принтера

Давно прошли те времена, когда для настройки принтера в Linux нужно было потратить полдня и результат при этом был неясен – принтер мог не заработать. Сейчас же все, как в Windows – включили принтер, подключили USB-кабель и Linux отображает уведомление о том, что она устанавливает принтер (рис. 11.1).

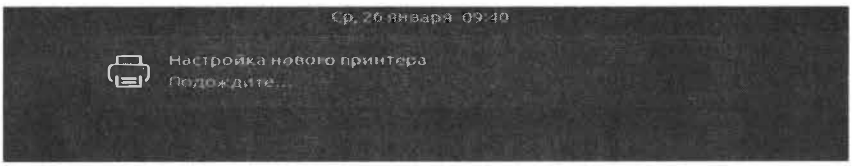


Рис. 11.1. Принтер найден и устанавливается

После этого уведомления вы должны увидеть другое уведомление – с названием модели принтера. Принтер устанавливается настолько быстро, что скриншот со вторым уведомлением я попросту не успел сделать. Поэтому идем в **Настройки**, **Принтеры** и смотрим, какой принтер был установлен (рис. 11.2).

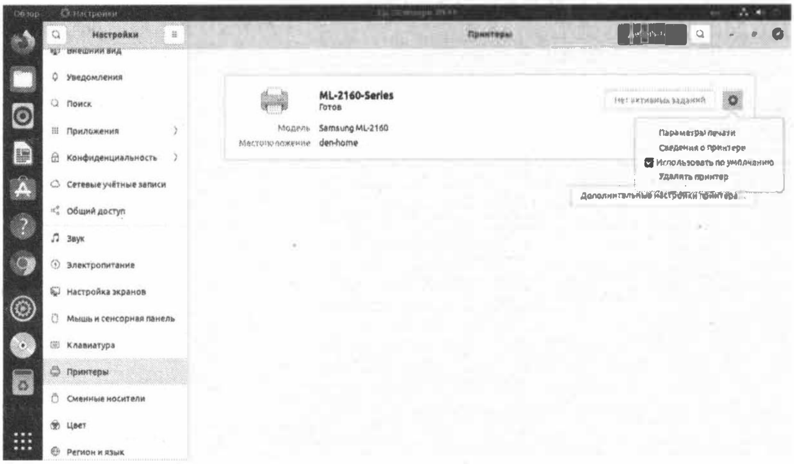


Рис. 11.2. Настройки => Принтеры

Здесь отображается производитель и модель принтера, а также его состояние (в данном случае – **Готов**). При нажатии на *шестеренку* открывается меню операций с принтером. Здесь можно назначить устройство принтером по умолчанию, если у вас несколько принтеров. Команда **Удалить принтер** позволяет удалить устройство, если оно вам не нужно. Команда **Параметры печати** открывает окно с настройками принтера (рис. 11.3).

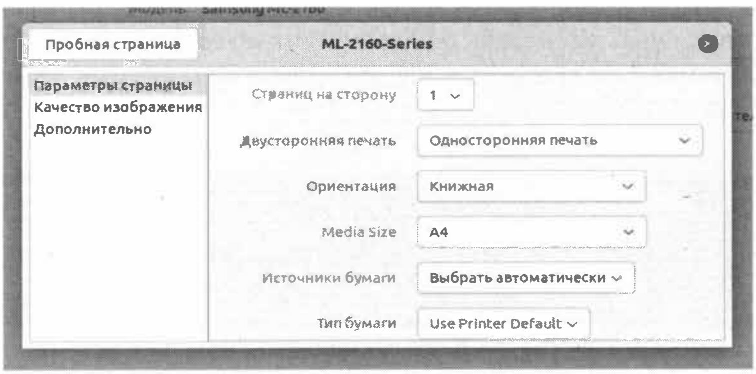


Рис. 11.3. Настройки принтера

Здесь нет ничего интересного – разве что можно выбрать размер бумаги, если вы хотите печатать не на стандартных листах А4. В разделе **Качество изображения** можно выбрать качество печати, если принтер поддерживает такую настройку (рис. 11.4).

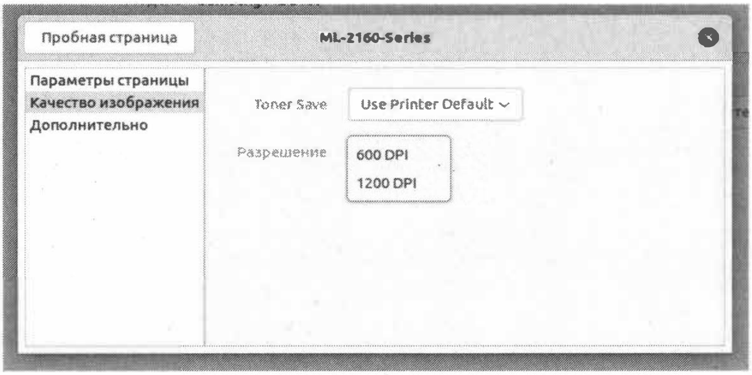


Рис. 11.4. Качество изображения

В дополнительных настройках самый полезный параметр – **Power Save**, задающий время неактивности принтера. По умолчанию принтер выключается (переходит в режим сна) через 5 минут, что не всегда удобно.



Рис. 11.5. Дополнительные настройки принтера

Нажмите кнопку **Дополнительные настройки принтера** – откроется окно **Принтеры**, в котором вы увидите список установленных принтеров в вашей системе. Щелкните правой кнопкой на принтере, чтобы открыть его контекстное меню. Выберите команду **Просмотр очереди печати** для просмотра очереди печати принтера. Здесь вы можете управлять заданиями на печать.

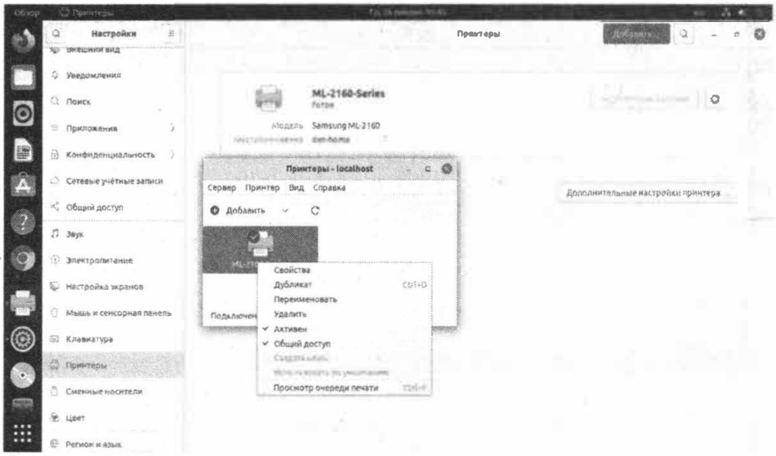


Рис. 11.6. Окно Принтеры

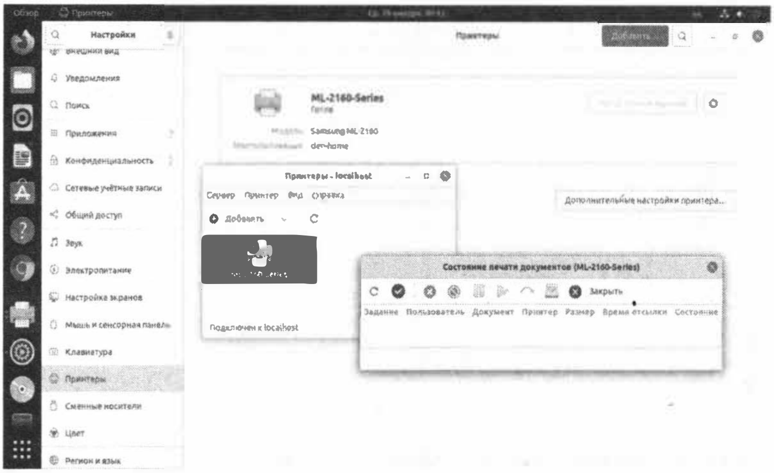


Рис. 11.7. Управление заданиями на печать

11.2. Осуществление печати из приложения

Как правило, в любой программе, поддерживающей печать, есть команда **Печать** и обычно она находится в меню **Файл**. На рис. 11.8 изображено окно печати приложения LibreOffice Writer – в нем есть область предварительного просмотра, позволяющее оценить результат печати. В большинстве случаев вам нужно просто нажать кнопку **Печать**.

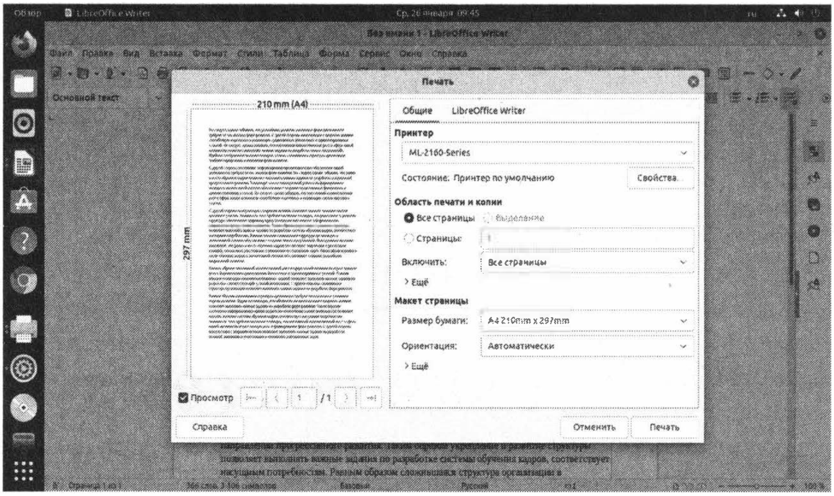
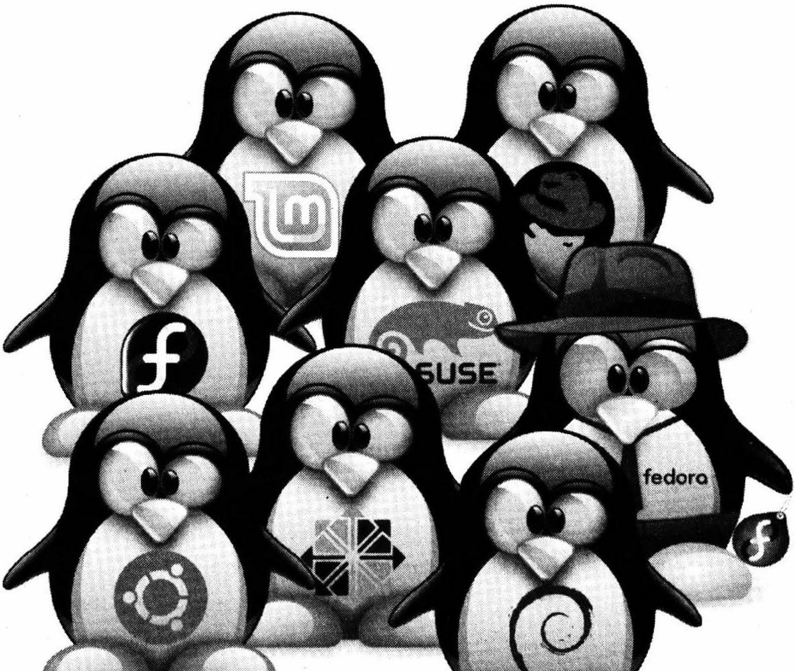


Рис. 11.8. Печать документа

Глава 12.

Файловая система



12.1. Какие файловые системы поддерживает Linux

Операционная система Linux поддерживает очень много операционных систем. Но самое главное – это модульный принцип организации ядра Linux. Даже если кто-то сегодня создаст файловую систему, о которой еще вчера никто ничего не знал, то ему достаточно создать модуль ядра и Linux будет поддерживать его операционную систему.

Родным для Linux является семейство файловых систем ext*. Самая древняя файловая система Linux называлась **ext** (сегодня вы вряд ли с ней столкнетесь), затем появились ext2, ext3 и ext4. Еще в 2010 году ходили слухи о ext5, но ее так и не создали.

Файловые системы ext3 и ext4 являются журналируемыми, то есть они ведут "журналы" своей работы, что позволяет произвести восстановление информации в случае сбоя. Журналы работают так: перед осуществлением операции файловая система записывает в журнал эту операцию, а после выполнения операции – удаляет запись из журнала. Если после занесения информации в журнал произошел сбой (например, отключение электричества), то после его устранения (подача электричества) файловая система выполнит все действия, которые она не успела выполнить. Конечно, это не панацея и резервные копии никто не отменял. Но все же лучше, чем ничего.

Однако не во всех дистрибутивах ext4 используется по умолчанию. Linux также поддерживает и другие файловые системы: XFS, ReiserFS, BtrFS, ZFS, JFS. Вы можете встретить дистрибутивы, в которых по умолчанию используется одна из этих файловых систем. У каждой из этих файловых систем есть свои отличия:

- **JFS** (Journaled File System) – 64-битная журналируемая файловая система созданная IBM, распространяется по лицензии GPL и благодаря этому факту она оказалась в Linux. Обладает высокой производительностью, но у нее маленький размер блока (от 512 байт до 4 Кб), поэтому на сервере данных ее можно использовать с большим успехом, но не на рабочих станциях, на которых производится обработка видео в реальном времени, так как размер блока для этих задач будет маловат. В отличие от ext3, в которую поддержка журнала была добавлена (по сути, ext3 – это то же самое, что и ext2, но с журналом), JFS была изначально журналируемой. Максимальный размер тома – 32 Пб, максимальный размер файла – 4 Пб.
- **ReiserFS** – самая экономная файловая система, поскольку позволяет хранить в одном блоке несколько файлов. В других файловых системах файл должен занимать как минимум 1 блок и получается, что если размер файла меньше размера блока, то "остаток" просто не используется. Когда в системе много небольших файлов, дисковое пространство используется очень нерационально. В ReiserFS все иначе. Если размер блока, скажем 4 Кб, то в него могут поместиться несколько файлов общим размером 4 Кб, а не только один, например, два файла по 2 Кб. Максимальный размер тома и файла зависят от версии ReiserFS и разрядности системы.
- **XFS** – высокопроизводительная (до 7 Гбайт/с) файловая система, разработанная Silicon Graphics. Изначально была рассчитана на большие размеры накопителей (более 2 Тб) и большие размеры файлов. Очень хорошо проявила себя при работе с файлами большого размера. Размер блока у этой файловой системы – от 512 байт до 64 Кбайта. Выделяет место экстентами (Extent — указатель на начало и число последовательных блоков). В экстентах выделяется место для хранения файлов, а также экстентами хранятся свободные блоки. Именно эта файловая система используется по умолчанию в современных версиях Fedora Server,
- **Btrfs (B-tree FS, "Better FS" или "Butter FS")** – файловая система, разработанная специально для Linux, и основанная на структурах Б-деревьев. Работает по принципу "копирование при записи" (copy-on-write). Создана компанией Oracle Corporation в 2007 году, распространяется по лицензии

GPL. Изначально планировалась как конкурент популярной файловой системе ZFS.

- **ZFS** (Zettabyte File System) – файловая система, созданная в Sun Microsystems для операционной системы Solaris. Позже она появилась и в Linux. Ее особенность – полный контроль над физическими и логическими носителями.

Кроме перечисленных выше файловых систем Linux поддерживает еще и файловые системы Windows – FAT, FAT32, NTFS. Также поддерживаются всевозможные сменные носители вроде оптических дисков, флешки, внешние жесткие диски и, соответственно, файловые системы на этих сменных носителях. На внешних жестких дисках используются файловые системы FAT32 или NTFS, если вы специально не переформатировали их в файловую систему Linux. На оптических дисках может использоваться UDF, ISO 9660, Joliet и подобные.

12.2. Какую файловую систему выбрать?

Файловых систем довольно много, так какую из них выбрать для вашей системы? На рабочих станциях и серверах общего назначения я бы использовал ext4 или ReiserFS. Последняя особенно хороша, если у вас много мелких файлов – тогда дисковое пространство будет использоваться более рационально.

На сервере баз данных лучше использовать JFS – тогда прирост производительности вам гарантирован. А вот XFS для некоторых видов серверов подходит так себе, однако, это не помешало разработчикам Fedora Server использовать эту файловую систему по умолчанию в своем дистрибутиве (начиная с версии 22). Видимо, повлияла высокая производительность и ориентация на большие объемы накопителей и файлов.

Очень неплохой является файловая система ZFS, особенно она хороша при управлении различными дисковыми устройствами. Вы можете создать пул и добавить в него несколько дисковых устройств. Этим ZFS чем-то похожа на LVM – менеджер логических томов.

Не спешите с выбором файловой системы именно сейчас. Может быть, при прочтении данной главы, вы найдете ответы на все ваши вопросы и остановите свой выбор на ext4.

Выбор файловой системы нужно производить даже не на основе характеристик самой файловой системы, а исходя из назначения компьютера. Например, для рабочей станции с головой хватит ext4. С сервером сложнее – нужно учитывать, какой это будет сервер. Как уже отмечалось, для сервера БД – JFS, для хостинга (где хранятся файлы других пользователей) – ReiserFS, если нужно работать с большими объемами данных – XFS.

12.3. Что нужно знать о файловой системе Linux

12.3.1. Имена файлов и каталогов

Нужно помнить следующие правила именования файлов и каталогов в Linux:

- Linux чувствительна к регистру символов, то есть файлы Document.txt и document.TXT – это разные документы.
- В Linux нет понятия "расширение" файла. Если в Windows мы привыкли, что последние символы после последней точки (обычно от 1 до 4 символов) называются расширением. В Linux такого понятия нет. Если кто-то и употребляет термин "расширение" в Linux, то это только для того, чтобы бывшим Windows-пользователям (которыми являемся, по сути, все мы) было понятнее, что имеется в виду.
- Максимальная длина имени файла – 254 символа.
- Имя может содержать любые символы (в том числе и кириллицу), кроме / \ ? < > * "|.
- Разделение элементов пути осуществляется с помощью символа /, а не \, как в Windows. В Windows мы привыкли к путям вида C:\Users\John, в Linux используется прямой слэш: /home/john.
- Если имя файла начинается с точки, он считается скрытым. Пример: .htaccess.

12.3.2. Файлы устройств

Уникальность файловой системы Linux в том, что для каждого устройства в Linux создается собственный файл в каталоге /dev. Загляните в каталог /dev – в нем вы найдете множество файлов для всех устройств вашей системы. Вот примеры некоторых файлов устройств:

- /dev/sda – первый жесткий диск, как правило, подключенный к первому SATA-контроллеру.
- /dev/sda1 – первый раздел на первом жестком диске. Нумерация разделов жестких дисков в Linux начинается с 1.
- /dev/mouse – файл устройства мыши.
- /dev/cpu – файл устройства процессора;
- /dev/cdrom – ваш CD/DVD-привод;
- /dev/random – файл устройства-генератора случайных чисел;
- /dev/tty1 – первая консоль (терминал).

Файлы устройств бывают двух типов: символьные, обмен информацией с которыми осуществляется посимвольно, и блочные, обмен информацией с которыми осуществляется блоками данных. Пример символьного устройства – /dev/ttyS0 – последовательный (COM) порт, пример блочного устройства – /dev/sda1 – раздел жесткого диска.

12.3.3. Корневая файловая система и основные подкаталоги первого уровня

Самое большое отличие, к которому придется вам привыкнуть – это наличие корневой файловой системы. Вспомните, как Windows управляет жесткими дисками. Представим, что у нас есть жесткий диск с двумя логическими дисками (разделами). Первый будет в Windows называться C:, а второй – D:. У каждого из этих логических дисков будет свой корневой каталог – C:\ и D:\.

В Linux все иначе. Представьте, что мы разбили жесткий диск /dev/sda на два раздела (как и в случае с Windows). Первый будет называться /dev/sda1 (Windows бы его назвала C:), а второй – /dev/sda2 (в Windows он был бы D:).

Мы установили Linux на первый раздел /dev/sda1. Точка монтирования этого раздела будет /, что соответствует корневой файловой системе. Второй раздел вообще никак не будет отображаться, пока вы его не *подмонтируете*. Подмонтировать можно к любому каталогу. Например, вы можете подмонтировать раздел /dev/sda2 к каталогу /home и тогда домашние каталоги пользователей будут храниться физически на другом разделе. Точка монтирования – это каталог, через который осуществляется доступ к другому разделу.

Правильнее сказать даже к другой файловой системе, которая физически может находиться на другом разделе, на другом жестком диске, на внешнем жестком диске, флешке и т.д.

Корневая файловая система содержит стандартные каталоги. У каждого каталога есть свое предназначение, например, в каталоге /bin хранятся стандартные программы, в каталоге /home – домашние каталоги пользователей, в каталоге /tmp – временные файлы и т.д. Назначение стандартных каталогов приведено в таблице 12.1.

Таблица 12.1. Назначение стандартных каталогов корневой файловой системы Linux

Каталог	Описание
/	Каталог корневой файловой системы
/bin	Содержит стандартные утилиты (cat, ls, cp и т.д.)
/boot	Содержит конфигурационный файл загрузчика и некоторые модули загрузчика
/dev	Содержит файлы устройств
/etc	В этом каталоге находятся конфигурационные файлы системы и программ
/home	Здесь хранятся домашние каталоги пользователей
/lib	Содержит библиотеки и модули
/lost+found	В этом каталоге хранятся восстановленные после некорректного размонтирования файловой системы файлы
/misc, /opt	Опциональные каталоги, могут содержать все, что угодно. Некоторые программы могут устанавливаться в каталог / opt
/media	Некоторые дистрибутивы монтируют сменные устройства (оптические диски, флешки) к подкаталогам этого каталога

/mnt	Содержит точки монтирования. Как правило, здесь хранятся стационарные точки монтирования, которые обычно описываются в файле /etc/fstab
/proc	Каталог псевдофайловой системы procfs
/root	Каталог пользователя *root*
/sbin	Содержит системные утилиты. Запускать эти утилиты имеет право только пользователь *root*
/tmp	Содержит временные файлы
/usr	Может содержать много чего – пользовательские программы (несистемные программы), документацию, исходные коды ядра и т.д.
/var	Содержит постоянно изменяющиеся данные системы – почтовые ящики, очереди печати, блокировки (locks) и т.д.

12.4. Ссылки

Ссылки позволяют одному и тому же файлу существовать в системе под разными именами. Ссылки бывают жесткими и символические. Сейчас разберемся в чем разница. Если на файл указывает хотя бы одна жесткая ссылка, вы не сможете его удалить. Количество ссылок на файл можно узнать командой *ls -l*. Что касается символических ссылок, то вы можете удалить файл, если на него указывает хоть 100 символических ссылок. После этого они будут "оборваны" – ссылки, как файлы, останутся на жестком диске, но они будут указывать на несуществующий файл.

У жестких ссылок есть одно ограничение. Они не могут указывать на файл, находящийся за пределами файловой системы. Представим, что каталог /tmp находится физически на одном и том же разделе, что и /. Тогда вы сможете создать ссылки на файлы, которые находятся в каталоге /tmp. Но если /tmp – это точка монтирования, к которой подмонтирован другой раздел, вы не сможете создать жесткие ссылки.

Для создания ссылок используется команда *ln*:

```
ln [-s] файл ссылка
```

Если параметр *-s* не указан, то будет создана ***жесткая*** *ссылка* на *файл*. Если параметр *-s* указан, то будет создана ***символическая*** *ссылка*.

12.5. Права доступа

12.5.1. Общие положения

В Linux, как и в любой многопользовательской системе, есть понятия владельца файла и прав доступа. Владелец – это пользователь, которому принадлежит файл. В большинстве случаев – это пользователь, создавший файл.

Права доступа определяют, кто и что может сделать с файлом. Права доступа файла может изменять владелец файла или пользователь *root*. Владелец может назначить, например, кто имеет право читать и изменять файл. Владелец также может "подарить" файл другому пользователю. После этого владельцем станет уже другой пользователь.

Права доступа у пользователя *root* максимальные, а это означает, что он может изменить владельца любого файла (вы можете создать файл, а *root* может сделать владельцем любого другого пользователя) и изменить права доступа любого файла. Пользователь *root* может удалить и изменить любой файл, может создать файл в любой папке и т.д. С одной стороны, это хорошо, но если злоумышленник завладеет паролем *root*, то хорошего в этой ситуации мало.

Права доступа в Linux по умолчанию настроены так, что пользователь владеет только своим домашним каталогом /home/<имя_пользователя>. Поэтому создавать файлы и выполнять другие операции по работе с файлами (удаление, редактирование, копирование и т.д.) пользователь может только в этом каталоге и то при условии, что файлы принадлежат ему.

Если в домашнем каталоге пользователя *root* создался файл, пользователь не сможет удалить или изменить его, поскольку он не является его владельцем. Сможет ли он прочитать этот файл, зависит от прав доступа к файлу (о них мы поговорим позже).

Остальные файлы, которые находятся за пределами домашнего каталога, пользователь может только просмотреть и то, если это не запрещено правами доступа. Например, файл /etc/passwd пользователь может просмотреть, а /etc/shadow – нет. Также пользователь не может создать файлы в корневой

файловой системе или в любом другом каталоге, который ему не принадлежит, если иное не установлено правами доступа к этому каталогу.

12.5.2. Смена владельца файла

Команда *chown* используется для изменения владельца файла/каталога. Формат такой:

```
chown <пользователь> <файл/каталог>
```

Здесь пользователь – это новый владелец файла. Чтобы подарить другому пользователю файл, вы должны быть или его владельцем, или пользователем *root*.

12.5.3. Определение прав доступа

Для изменения прав доступа используется команда *chmod*. Для изменения прав доступа вы должны быть владельцем файла/каталога или же пользователем *root*. Формат команды следующий:

```
chmod <права> <файл/каталог>
```

Права доступа состоят из трех наборов: для владельца, для группы владельца и для прочих пользователей. Первый набор задает возможности владельца файла, второй – для группы пользователей, в которую входит владелец и третий – для всех остальных пользователей.

В наборе может быть три права – чтение (r), запись (w) и выполнение (x). Для файла право на выполнение означает возможность запустить этот файл на выполнение (обычно используется для программ и сценариев). Право выполнения для каталога – возможность просматривать этот каталог.

Права доступа в наборе определяются четко в определенном порядке и могут быть представлены, как символьном, так и числовом виде (в двоичной или восьмеричной системе). Рассмотрим несколько наборов прав доступа:

- 100 – только чтение

- 110 – чтение и запись
- 101 – чтение и выполнение
- 111 – чтение, запись, выполнение

Учитывая, что права доступа задаются для владельца, группы и остальных пользователей, полный набор прав доступа может выглядеть так:

```
111 100 000
```

В этом наборе мы предоставляем полный доступ (в том числе и выполнение) владельцу, группе владельца разрешено только чтение, остальным пользователям доступ к файлу/каталогу вообще запрещен.

В двоичной системе права доступа мало кто записывает. В основном их преобразуют в восьмеричную систему. Если вы забыли, то вам поможет следующая таблица (табл. 12.2).

Таблица 12.2. Преобразование из двоичной в восьмеричную систему

Двоичная система	Восьмеричная система
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

Если вы видите право доступа 666, то никакой дьявольщины в нем нет, это всего лишь полный доступ к обычному файлу (не к программе и не к сценарию). Для каталога полные права доступа выглядят как 777 – чтение, изменение и просмотр каталога для владельца, группы и прочих пользователей.

Просмотреть текущие права доступа можно командой `ls -l <файл/каталог>`, например:

```
# ls -l config

-rw-r--r--. 1 root root 110375 янв 2 08:28 config
```

Как мы видим, задано три набора rw-, r--, r--. Выходит, владельцу разрешена запись и чтение файла, остальным пользователям (группа и прочие) – только чтение. В восьмеричной системе этот набор прав доступа выглядит как 644.

Первый символ (в нашем случае это -) является признаком каталога. Если бы мы выводили права доступа каталога, то вместо - здесь был бы символ **d**. Для файла выводится просто "-".

Символьный способ задания прав доступа немного проще, но лично я предпочитаю числовой. Рассмотрим, как использовать символьный:

```
# chmod +x config
```

Просмотрим опять права доступа:

```
# ls -l config

-rwxr-xr-x. 1 root root 110375 sep 2 08:28 config
```

Как видите, право выполнение было добавлено во все три набора прав доступа.

12.5.4. Специальные права доступа

В Linux есть еще специальные права доступа SUID (Set User ID root) и SGID (Set Group ID root), позволяющие обычным пользователям запускать программы, которые требуют для своей работы прав *root*.

В современных дистрибутивах Linux вам придется изменять эти права доступа чрезвычайно редко (может быть даже вообще никогда), но вам нужно знать, как их изменить. Например, если программу /usr/sbin/program вы хотите разрешить запускать с правами *root* обычным пользователям, установите права доступа так:

```
# chmod u+s /usr/sbin/program
```

Использование SUID – плохое решение с точки зрения безопасности. Правильнее использовать команду *sudo*, если какому-то пользователю будут нужны права *root*.

12.6. Атрибуты файла

В Linux кроме прав доступа есть еще и атрибуты файла, подобно атрибутам файла в других операционных системах. Изменить атрибуты файла можно командой *chattr*:

```
chattr +/-<атрибуты> <файл>
```

Просмотреть установленные атрибуты можно командой *lsattr*:

```
lsattr <файл>
```

Некоторые полезные атрибуты файлов приведены в таблице 12.3.

Таблица 12.3. Полезные атрибуты файлов

Атрибут	Описание
i	Запрещает изменение, переименование и удаление файла. Этот атрибут можно установить для критических конфигурационных файлов или для каких-либо других критических данных. Установить (как и сбросить) этот атрибут может только пользователь *root* или процесс с возможностью CAP_LINUX_IMMUTABLE. Другими словами, сбросить этот атрибут просто так нельзя – нужны только права *root*
u	При удалении файла с установленным атрибутом u его содержимое хранится на жестком диске, что позволяет легко восстановить файл

c	Файл будет сжиматься. Можно установить этот атрибут для больших файлов, содержащих несжатые данные. Доступ к сжатым файлам будет медленнее, чем к обычным, поэтому плохое решение устанавливать этот атрибут для файлов базы данных. Этот атрибут нельзя устанавливать для файлов, уже содержащих сжатые данные – архивы, JPEG-фото, MP3/MP4-файлы и т.д. Этим вы не только не уменьшите его размер, но и замедлите производительность
S	Данные, записываемые в файл, сразу будут сброшены на диск. Аналогично выполнению команды *sync* сразу после каждой операции записи в файл
s	Прямо противоположен атрибуту *u*. После удаления файла, принадлежащие ему блоки будут обнулены и восстановить их уже не получится

Пример установки атрибута:

```
# chattr +i config
```

Пример сброса атрибута:

```
# chattr -i config
```

12.7. Поиск файлов

Для поиска файлов вы можете использовать команды *which*, *locate* и *find*. Первая используется только для поиска программ. Она позволяет определить, в каком каталоге находится исполнимый файл той или иной программы, например:

```
# which pppd
/sbin/pppd
```

Данную программу очень удобно использовать администратору, когда нужно вычислить месторасположение программы, например, чтобы указать точный путь к программе в каком-то сценарии или конфигурационном файле.

Команда *locate* позволяет произвести быстрый поиск файла. Однако команда *locate* будет работать не во всех дистрибутивах, а только там, где доступен *updatedb*, который и формирует базу данных, по которой производит поиск команда locate. Если файл будет на диске, но его не будет в базе данных, то *locate* его не найдет – вот в чем основной недостаток этой команды. Для обновления базы данных нужно ввести команду *updatedb* (или дождаться, пока планировщик обновит базу данных).

Преимущество команды *locate* в том, что поиск файла производится практически мгновенно, особенно по сравнению с командой *find*. Однако если файла не будет в базе данных (файл был создан после обновления базы данных *locate*), команда *locate* его не найдет.

Конфигурация *updatedb* хранится в файле /etc/updatedb.conf (листинг 12.1).

Листинг 12.1. Файл /etc/updatedb.conf

```
PRUNE_BIND_MOUNTS = "yes"
PRUNEFS = "9p afs anon_inodefs auto autofs bdev binfmt_misc cgroup cifs coda configfs
cpuset debugfs devpts ecryptfs exofs fuse fuse.sshfs fusectl gfs gfs2 hugetlbfs
inotifyfs iso9660 jffs2 lustre mqueue ncpfs nfs nfs4 nfsd pipefs proc ramfs rootfs
rpc_pipefs securityfs selinuxfs sfs sockfs sysfs tmpfs ubifs udf usbfs"
PRUNENAMES = ".git .hg .svn"
PRUNEPATHS = "/afs /media /mnt /net /sfs /tmp /udev /var/cache/ccache
/var/lib/yum/yumdb /var/spool/cups /var/spool/squid /var/tmp"
```

Если параметр PRUNE_BIND_MOUNTS равен *yes*, файловые системы, смонтированные в режиме *bind* не исследуются при помощи *updatedb*. Параметр PRUNEFS задает типы файловых систем, которые не будут исследоваться *updatedb*. Аналогично, параметры PRUNENAMES и PRUNEPATHS задают имена файлов (у нас заданы "расширения") и пути (каталоги).

В листинге 12.1 приведен пример файла update.conf по умолчанию из Fedora Server. Вы можете отредактировать его под свои нужды, например, закомментировать параметр PRUNENAMES, отредактировать параметр PRUNEPATHS и т.д.

Теперь перейдем к третьей и самой универсальной команде поиска – *find*. Формат вызова следующий:

```
$ find список_поиска выражение
```

Полное описание команды *find* вы найдете в справочной системе (команда *man mount*), а мы рассмотрим несколько примеров.

```
$ find / -name test.txt
```

Мы ищем все файлы с именем test.txt, начиная с корневого каталога /. Если нужно найти все текстовые файлы (*.txt), начиная с корневого каталога, тогда команда будет такой:

```
$ find / -name '*.txt'
```

Следующая команда ищет только пустые файлы (параметр *-empty*):

```
$ find . -empty
```

Если нужно задать размер файла, тогда можем указать размер явно. В следующем примере мы задаем размер файла – от 500 до 700 Мб:

```
$ find ~ -size +500M -size -700M
```

Команда *find* может не только находить файлы, но и выполнять действие для каждого найденного файла. В следующем примере мы находим все старые резервные копии ("расширение" .bak) и удаляем их:

```
# find / -name *.bak -ok rm {} \;
```

Поиск с помощью *find* занимает немало времени – ведь нет никакой базы данных. Команде *find* нужно "пройтись" по всем каталогам и проверить в них наличие искомых файлов.

12.8. Монтирование файловых систем

12.8.1. Монтируем файловые системы вручную

Ранее было сказано, что такое точка монтирования – это каталог, через который происходит доступ к файловой системе, физически размещенной на другом носителе (другом разделе жесткого диска, флешке, оптическом диске или даже на другом компьютере).

Для монтирования файловой системы используется команда *mount*, для размонтирования – *umount*:

```
# mount [опции] <имя устройства> <точка монтирования>
# umount <имя устройства или точка монтирования>
```

Для монтирования файловой системы нужны права *root*, поэтому команды *mount* и *umount* нужно вводить с правами *root*.

Представим, что мы подключили флешку. Если у вас один жесткий диск (/dev/sda), то флешке будет назначено имя /dev/sdb, если жестких дисков два, то флешке будет назначено следующее имя – /dev/sdc и т.д.

На одном носителе (это касается жестких дисков, флешек и подобных носителей) может быть несколько разделов, которым назначаются номера, нумерация начинается с единицы. Поэтому вы не можете подмонтировать все устройство /dev/sdc. Вы должны указать номер раздела.

Подмонтируем нашу флешку (пусть это будет устройство /dev/sdc и на нем будет всего один раздел с номером 1):

```
# mount /dev/sdc1 /mnt/usb
```

Каталог /mnt/usb – это и есть точка монтирования. Точка монтирования должна существовать до вызова команды *mount*, то есть вы не можете подмонтировать файловую систему к несуществующему каталогу.

После этого вы можете обращаться к файлам и каталогам на флешке через каталог /mnt/usb:

```
# ls /mnt/usb
```

Итак, последовательность действий такая: создание точки монтирования (один раз), монтирование файловой системы, работа с файловой системой и размонтирование. Размонтирование осуществляется командой *umount*. В качестве параметра команды *umount* нужно передать или название точки монтирования или имя устройства:

```
# mkdir /mnt/usb
# mount /dev/sdc1 /mnt/usb
# cp test.txt /mnt/usb
# umount /mnt/usb
```

Очень важно размонтировать файловую систему, особенно это касается внешних файловых систем. При завершении работы система автоматически размонтирует все смонтированные файловые системы.

Думаю, в общих чертах операция монтирования должна быть понятной. Теперь поговорим о параметрах команды *mount*. Самый часто используемый параметр – это параметр *-t*, позволяющий задать тип монтируемой файловой системы. Обычно команда *mount* сама в состоянии распознать тип файловой системы, но в некоторых случаях ей нужно указать его вручную. Вот наиболее распространенные типы файловых систем:

- **vfat** – файловая система Windows (FAT/FAT32);
- **ntfs** – файловая система Windows NT;
- **ntfs-3g** – драйвер ntfs-3g для чтения и записи NTFS (рекомендуется);
- **ext2/ext3/ext4** – различные версии файловой системы Linux;
- **iso9660** – файловая система оптического диска CD/DVD;
- **udf** – иногда Windows форматирует оптический диск как UDF;
- **reiserfs** – файловая система ReiserFS;
- **smbfs** – файловая система Samba;
- **nfs** – сетевая файловая система.

Например, в случае с NTFS рекомендуется использовать драйвер ntfs-3g:

```
# mount -t ntfs-3g /dev/sdc1 /mnt/usb
```

Параметр *-r* позволяет смонтировать файловую систему в режиме "только чтение", параметр *-w* монтирует файловую систему в режиме "чтение/запись", но обычно в этом режиме файловая система монтируется по умолчанию, поэтому в нем нет необходимости.

Параметр *-a* монтирует все файловые системы, перечисленные в файле /etc/fstab, за исключением тех, для которых указана опция *noauto*.

12.8.2. Имена устройств

Интерфейсов жестких дисков довольно много – IDE (ATA/PATA), SATA (Serial ATA), SCSI, USB. Раньше жесткие диски с интерфейсом IDE назывались в Linux /dev/hd? (? – буква, которая зависит от того, как подключен жесткий диск). Жесткие диски с интерфейсом SATA и SCSI назывались /dev/sd? (? – буква диска, соответствующая его порядковому номеру при подключении к интерфейсу).

Сейчас даже IDE-диски называются /dev/sd? (как и SATA/SCSI), что сначала вносило некую путаницу. Но жесткие диски с интерфейсом IDE вышли из моды и практически не используются. Мода на SCSI-диски также практически закончилась, поскольку SATA-диски такие же быстрые, как и SCSI – некоторые обеспечивают такую же производительность, как и SCSI – в некоторых случаях чуть меньше, в некоторых – даже больше. Так что SCSI уже можно списывать со счета – если вам достался сервер со SCSI-диском, отказываться от него не стоит, а вот новый сервер будет поставляться или с интерфейсом SATA или с интерфейсом SAS.

Интерфейс SAS (Serial Attached SCSI) обратно совместим с интерфейсом SATA и позволяет последовательно подключать SATA-диски и обеспечивает пропускную способность в 6 Гбит/с. Если вы купите сервер с интерфейсом SAS, то в большинстве случаев он будет оснащен высокопроизводительными SATA-дисками, а не SCSI-дисками. Поэтому никакой путаницы уже нет.

Что же касается USB-дисков (флешки и внешние жесткие диски), то они также получают обозначение /dev/sd?.

Оптические диски (приводы CD/DVD) в большинстве случаев называются /dev/sr?, где ? – номер привода, нумерация начинается с 0.

Чтобы узнать, какие жесткие диски и оптические приводы установлены в вашем компьютере, введите команды (рис. 12.1):

```
ls /dev/sd?
ls /dev/sr0
```

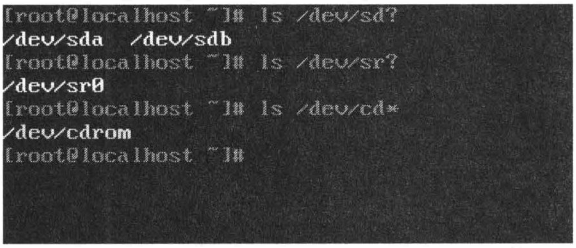


Рис. 12.1. Жесткие и оптические диски

Если у вас один оптический диск, можно смело использовать ссылку /dev/cdrom. Из рис. 12.1 видно, что у нас установлено два жестких диска – /dev/sda и /dev/sdb, а также один оптический привод /dev/sr0.

Для монтирования не достаточно указать имя всего устройства, нужно уточнить номер раздела. Когда разделов много, вы можете забыть (или не знать), какой именно раздел вам нужен. Вы можете использовать команду *fdisk*: запустите *fdisk <имя устройства>*, а затем введите команду *p* для вывода таблицы разделов и команду *q* для выхода из *fdisk*.

Посмотрите рис. 12.2. Из него становится понятно, что на нашем жестком диске есть два раздела – /dev/sda1 и /dev/sdb2.

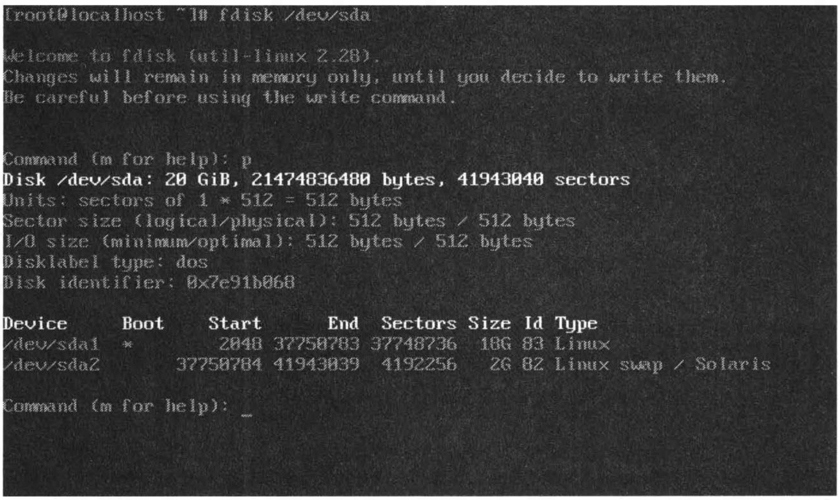


Рис. 12.2. Программа fdisk

Кроме коротких имен вроде /dev/sd? в современных дистрибутивах часто используются идентификаторы UUID. Загляните в файл /etc/fstab и в нем в большинстве случаев вместо привычных имен /dev/sd? вы обнаружите вот такие "страшные" имена:

```
# В Fedora, Debian, Ubuntu
UUID=2f149af9-3bff-44bd-d16s-ff98s9a7116d / ext4 defaults 0  1

# openSUSE
/dev/disk/by-id/dm-name-suse-server-root / ext4  defaults 1 1
```

Преимущество идентификаторов UUID в том, что они не изменяются, если вы иначе подключите жесткий диск. Представим, что у вас есть два жестких диска – /dev/sda и /dev/sdb. На первый вы установили Linux (в раздел /dev/sda1), а второй используете для хранения данных (на нем всего один раздел /dev/sdb1, который монтируется, как /home). Вы отключили оба диска, а затем, подключая, перепутали их местами. В итоге второй диск стал диском /dev/sda, а первый – /sdb. При загрузке может случится конфуз, точнее, система вообще не загрузится. Даже если вы выберете в BIOS SETUP загрузку со второго жесткого диска, тоже ничего хорошего не выйдет. С UUID достаточно выбрать загрузку со второго жесткого диска и система будет загружена.

Вы можете использовать обычные стандартные имена, а можете использовать UUID-идентификаторы. Узнать, какие UUID-идентификаторы соответствуют каким обычным именам, можно с помощью команды:

```
ls -l /dev/disk/by-uuid/
```

Вывод этой команды представлен на рис. 12.3.

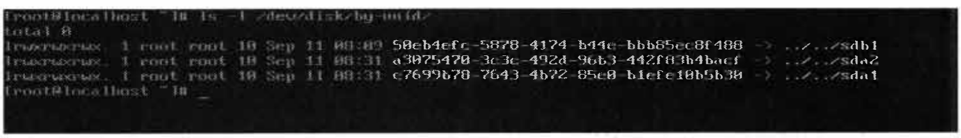


Рис. 12.3. Соответствие UUID-идентификаторов обычным именам

12.8.3. Монтируем файловые системы при загрузке

Вводить команды *mount* при каждой загрузке не очень хочется, поэтому проще "прописать" файловые системы в файле /etc/fstab, чтобы система смонтировала их при загрузке.

Формат файла /etc/fstab следующий:

```
устройство точка тип опции флаг_копирования флаг_проверки
```

Первое поле – это устройство, которое будет монтироваться к точке монтированию – второе поле. Вы можете использовать, как обычные имена, так и UUID. Третье поле – тип файловой системы. Четвертое поле – параметры файловой системы (табл. 12.4), последние два поля – это флаг резервной копии и флаг проверки. Первый флаг определяет, будет ли файловая система заархивирована командой *dump* при создании резервной копии (1 – будет, 0 – нет). Второй флаг определяет, будет ли файловая система проверяться программой *fsck* на наличие ошибок (1, 2 – будет, 0 – нет). Проверка производится, если достигнуто максимальное число попыток монтирования для файловой системы или если файловая система была размонтирована некорректно. Для корневой файловой системы это поле должно содержать 1, для остальных файловых систем – 2.

Таблица 12.4. Параметры файловой системы

Опция	Описание
defaults	Параметры по умолчанию
user	Разрешает обычному пользователю монтировать/размонтировать данную файловую систему
nouser	Запрещает обычному пользователю монтировать/размонтировать данную файловую систему. Смонтировать эту ФС может только *root*. Используется по умолчанию
auto	ФС будет монтироваться при загрузке. Используется по умолчанию, поэтому указывать не обязательно
noauto	ФС не будет монтироваться при загрузке системы

exec	Разрешает запуск исполнимых файлов на данной ФС. Используется по умолчанию. Для Windows-файловых систем (vfat, ntfs) рекомендуется использовать опцию *noexec*
noexec	Запрещает запуск исполнимых файлов на данной ФС
rw	ФС будет монтироваться в режиме "только чтение"
rw	ФС будет монтироваться в режиме "чтение запуск". По умолчанию для файловых систем, которые поддерживают запись
utf8	Для преобразования имен файлов будет использоваться кодировка UTF8
data	Задает режим работы журнала (см. ниже)

12.8.4. Автоматическое монтирование файловых систем

В современных дистрибутивах Linux сменные носители вроде USB-дисков и оптических дисков монтируются автоматически:

- Debian, Ubuntu, Fedora, CentOS – монтирование производится к каталогу /media/<метка_устройства>. В качестве метки может использоваться или метка, установленная при форматировании, или серийный номер устройства, если метка не устанавливалась.
- openSUSE – монтирование будет производиться к каталогу /var/run/media/<имя пользователя>/<метка>.

За автоматическое монтирование отвечает демон *automount*, который вы можете отключить, если автоматическое монтирование вам не нужно.

12.9. Работа с журналом

Существует три режима работы журналируемой файловой системы ext3/ext4: *journal*, *ordered* и *writeback*. По умолчанию используется режим *ordered* – оптимальный баланс между производительностью и надежностью. В этом

режиме в журнал будет заноситься информация только об изменении метаданных.

Самый медленный режим *journal*. В этом режиме в журнал записывается максимум информации, которая понадобится при восстановлении в случае сбоя. Режим очень медленный и использовать его следует только, если безопасность для вас важнее, чем производительность.

Самый быстрый режим *writeback*, но в нем, по сути, журнал не будет использоваться и у вас не будет никакой защиты, например, от той же перезагрузки.

Режим работы журнала задается параметром *data*, например:

```
/dev/sdb1 /home ext4 data=journal 1 2
```

12.10. Преимущества файловой системы ext4

Поговорим о преимуществах файловой системы ext4. Возможно, она вас полностью устроит и вам не придется искать другую файловую систему для своего компьютера.

Впервые файловая система ext4 появилась в ядре версии 2.6.28. По сравнению с ext3, максимальный размер раздела был увеличен до 1 эксбибайта (1024 петабайтов), а максимальный размер файла составляет 2 Тб. По производительности новая файловая система ext4 превзошла файловые системы ext3, Reiserfs, XFS и Btrfs (в некоторых операциях).

Так, ext4 опередила знаменитую XFS в тесте на случайную запись. Файловая система Btrfs провалила этот тест с огромным "отрывом" от лидеров – XFS и ext4. Производительность ext4 была примерно такой же, как у XFS, но все-таки немного выше, чем у XFS. В Интернете вы найдете множество тестов производительностей – просмотрите их, если вам интересно.

Основной недостаток ext3 заключается в ее методе выделения места на диске. Ее способ выделения дискового пространства не отличается производительностью, а сама файловая система эффективна для небольших файлов, но никак не подходит для хранения огромных файлов. В ext4 для более эффективной организации данных используются экстенты. Экстент – это непрерывная область носителя информации. К тому же ext4 откладывает выделение дискового пространства до последнего момента, что еще более увеличивает производительность.

Файловая система ext3 может содержать максимум 32 000 каталогов, в ext4 количество каталогов не ограничено.

В журнале ext4 тоже произошли изменения – в журнале ext4 используются контрольные суммы, что повышает надежность ext4 по сравнению с ext3.

Выходит, по сравнению с ext3, у ext4 есть следующие преимущества:

- Улучшена производительность – производительность почти достигла XFS, а в некоторых тестах даже превышает ее.
- Улучшена надежность – используются контрольные суммы журналов.
- Улучшена масштабируемость – увеличен размер раздела, размер файла и поддерживается неограниченное количество каталогов.

12.11. Специальные операции с файловой системой

12.11.1. Монтирование NTFS-разделов

Не думаю, что на сервере вам придется монтировать NTFS-разделы, но ситуации бывают разные. Для монтирования NTFS-раздела используется модуль ntfs-3g, который в большинстве случаев уже установлен по умолчанию. Если он не установлен, для его установку введите команду (замените *yum* на имя вашего менеджера пакетов):

```
# yum install ntfs-3g
```

Команда монтирования NTFS-раздела выглядит так:

```
# mount -t ntfs-3g раздел точка_монтирования
```

Например, вам кто-то принес флешку, отформатированную как NTFS. Для ее монтирования введите команду (измените только имя устройства и точку монтирования):

```
# mount -t ntfs-3g /dev/sdb1 /mnt/usb
```

Модуль ntfs-3g выполняет монтирование в режиме чтение/запись, поэтому вы при желании можете произвести запись на NTFS-раздел.

12.11.2. Создание файла подкачки

При нерациональном планировании дискового пространства может возникнуть ситуация, когда раздела подкачки стало мало или вы вообще его не создали. Что делать? Повторная разметка диска требует времени, а выключать сервер нельзя. Сервер тормозит, поскольку ему не хватает виртуальной памяти, а ждать от начальства подписи на дополнительные модули оперативной памяти придется еще неделю. А за это время пользователи вас окончательно достанут своими жалобами.

Выход есть. Он заключается в создании файла подкачки на жестком диске. Такой файл подкачки будет работать чуть медленнее, чем раздел подкачки, но это лучше, чем вообще ничего. Хотя, если у вас SSD-диск, никакой разницы в производительности практически не будет.

Первым делом нужно создать файл нужного размера. Следующая команда создает в корне файловой системы файл swap01 размером 1 Гб:

```
# dd if=/dev/zero of=/swap01 bs=1k count=1048576
```

После этого нужно создать область подкачки в этом файле:

```
# mkswap /swap01 1048576
```

Наконец, чтобы система "увидела" файл подкачки, его нужно активировать:

```
# swapon /swap01
```

Чтобы не вводить эту команду после каждой перезагрузке сервера, нужно обеспечить ее автоматический запуск.

12.11.3. Файлы с файловой системой

Только что было показано, как создать файл произвольного размера, а потом использовать его в качестве файла подкачки. При желании этот файл можно

отформатировать, как вам угодно. Даже можно создать в нем файловую систему.

Рассмотрим небольшой пример. Давайте опять создадим пустой файл размером 1 Гб:

```
# dd if=/dev/zero of=/root/fs01 bs=1k count=1048576
```

После этого нужно создать файловую систему в этом файле:

```
# mkfs.ext3 -F /root/fs01
```

Чтобы не заморачиваться, я создал самую обычную файловую систему ext3. После этого файл с файловой системой можно подмонтировать и использовать как обычных сменный носитель, то есть записывать на него файлы:

```
# mkdir /mnt/fs01
# mount -t ext3 -o loop /root/fs01 /mnt/fs01
```

После того, как закончите работу с файлом, его нужно размонтировать:

```
# umount /mnt/fs01
```

Зачем это вам нужно – знаете только вы. В конце-концов, можно использовать или зашифрованную файловую систему или просто архив. Но для общего развития это очень важно, особенно, если вы надумаете создавать собственный дистрибутив Linux.

12.11.4. Создание и монтирование ISO-образов

Все мы знаем утилиты, позволяющие в Windows подмонтировать ISO-образ диска. В Linux все подобные операции делаются с помощью штатных средств и никакие дополнительные программы не нужны.

Представим, что нам нужно создать образ диска. Если диск вставлен в привод, для создания его ISO-образа выполните команду:

```
$ dd if=/dev/cdrom of=~/dvd.iso
```

Здесь, /dev/cdrom – имя устройства (в Linux это имя соответствует любому оптическому приводу – CD или DVD), а dvd.iso – файл образа.

Иногда ставится другая задача: есть папка, по которой нужно создать ISO-образ. То есть у нас диска, но есть файлы, которые нужно записать на диск, но прежде вы хотите создать его ISO-образ.

Пусть у нас есть папка ~/dvd и нужно создать ISO-образ, содержащий все файлы из этой папки. Файл образа будет опять называться ~/dvd.iso. Для этого используйте команду *mkisofs*:

```
$ mkisofs -r -jcharset utf8 -o ~/dvd.iso ~/dvd
```

Чтобы проверить, что образ был создан корректно, его нужно подмонтировать к нашей файловой системе:

```
# mkdir /mnt/iso-image
# mount -o loop -t iso9660 dvd.iso /mnt/iso-image
```

Здесь все просто: опция *-o loop* означает, что будет монтироваться обычный файл, а не файл устройства, опция *-t* задает тип файловой системе, далее следуют название файла и название папки, к которой будет выполнено монтирование.

12.12. Файлы конфигурации Linux

12.12.1. Содержимое каталога /etc

Думаю, среди читателей этой книги, нет таких, которые бы не знали о реестре Windows. В реестре Windows хранятся настройки самой системы и практически всех программ (исключения составляют лишь устаревшие программы, хранящие свои параметры в ini-файлах).

Так вот, в Linux есть свое подобие реестра – это каталог /etc. И вы можете быть уверены, что в /etc находятся все настройки системы и всех программ.

Да, некоторые программы также хранят персональные настройки в домашних каталогах пользователя, но глобальные настройки обычно хранятся в каталоге /etc.

Преимущество каталога /etc перед реестром – в том, что для его редактирования вам не нужен какой-то особый редактор. В каталоге /etc просто хранится набор текстовых конфигурационных файлов. Вы можете просмотреть или отредактировать файл с помощью любого текстового редактора. Также можно легко скопировать файлы/каталоги, например, перед изменением. Единственный недостаток – вам нужно знать формат каждого файла, но обычно с этим проблем никаких нет, поскольку файлы снабжены подробными комментариями, путь и на английском языке.

12.12.2. Конфигурационные файлы

Чтобы эффективно настраивать систему, нужно ориентироваться в содержимом конфигурационного каталога. Если загляните в /etc, то кроме подкаталогов в нем вы обнаружите и файлы, которые находятся непосредственно в /etc. Описание этих файлов приведено в таблице 12.5. Содержимое каталога может отличаться в зависимости от используемого дистрибутива и установленного программного обеспечения. Но общая картина будет примерно одинаковой во всех дистрибутивах.

Таблица 12.5. Файлы из каталога /etc

Файл	Описание
DIR_COLORS	Конфигурация цвета для утилиты **ls**. Здесь вы можете указать, каким цветом будут выводиться каталоги, файлы, ссылки и т.д.
GREP_COLORS	Конфигурация цвета для утилиты **grep**
adjtime	Содержит параметры для корректировки аппаратных часов
aliases	База данных псевдонимов для почтовых агентов (MTA)

at.deny	Содержит пользователем, которым запрещено использовать планировщик **at**. Дополнительная информация доступна в **man at**
anacrontab	Конфигурация (таблица расписания) планировщика *anacron*
bash.bashrc	Глобальный файл конфигурации оболочки **bash**. Локальные файлы конфигурации находятся в домашнем каталоге каждого пользователя
bind.keys	Содержит ключи для DNS-сервера bind9
bindresvport.blacklist	Содержит список номеров портов в диапазоне от 600 до 1024, которые не могут быть использованы *bindresvport*, который обычно вызывается IRC-службами. По умолчанию запрещены порты 623, 631, 636, 664, 774, 921, 993 и 995, которые используются различными сетевыми службами.
cron.deny	Список пользователей, которым запрещено использовать *cron*
crontab	Таблица расписания демона *crond*
*crony**	Конфигурация NTP (сервер времени)
csh.cshrc	Файл конфигурации для оболочки C Shell
csh.login	Глобальный файл конфигурации для C Shell. Определяет поведения во время регистрации пользователя (login) в системе. На самом деле – это сценарий, который запрещено редактировать вручную. Он должен изменяться только время обновления системы. Вместо этого лучше редактировать /etc/csh.login.local, если вам нужно внести изменения в настройки вашего локального окружения
crypttab	Содержит информацию о зашифрованных томах

defaultdomain	Содержит доменное имя для сервисов NIS и NIS+
dhclient.conf	Файл конфигурации для DHCP-клиента
dhclient6.conf	Файл конфигурации для DHCP-клиента, версия IPv6
dhcpd.conf	Файл конфигурации для DHCP-сервера
dhcpd6.conf	Файл конфигурации для DHCP-сервера, версия IPv6
dialogrc	Конфигурационный файл для пакета dialog, определяет цветовую схему диалоговых интерфейсов, построенных с помощью пакета dialog
dnsmasq.conf	Конфигурационный файл для dnsmasq (DNS-маскарадинга)
dracut.conf	Содержит параметры dracut – средства, которое формирует initramfs
drirc	Конфигурационный файл для репозитария DRI CVS
environment	Файл используется PAM-модулем pam_env. Содержит переменные окружения, описанные в виде пар КЛЮЧ=ЗНАЧЕНИЕ, по одной паре в одной строке
esd.conf	Параметры EsounD (Enlightened Sound Daemon), который используется для смешивания вместе некоторых цифровых аудио потоков для проигрывания на одиночном устройстве
ethers	Содержит 48-битные Ethernet-адреса и соответствующие им IP-адреса или имена узлов. Может использоваться некоторыми сетевыми службами для разрешения MAC-адресов в IP-адреса
exports	Содержит список экспортируемых файловых систем

fedora-release	Информация о релизе Fedora
filesystems	Информационный файл, содержит некоторые характеристики и атрибуты файловых систем. В нем нет ничего интересного
fstab	Содержит список файловых систем, которые будут монтироваться автоматически при загрузке системы
ftpusers	Список пользователей, которые НЕ могут войти в систему по FTP. Среди них вы найдете пользователя *root* и многие другие системные учетные записи, которые используются для сетевых сервисов и обычно обладают повышенными привилегиями
group	Содержит группы пользователей
host.conf	Задает порядок разрешения доменных имен
hostname	Содержит доменное имя узла
hosts	Ранее использовался для разрешения IP-адресов в доменные имена. Сейчас для этого используется система DNS, но вы все равно можете определить в нем некоторые IP-адреса, если ваша сеть не использует DNS или же вам нужно переопределить разрешение для определенного IP-адреса. Обе ситуации в наше время настолько редки, что похожи на что-то из области технической фантастики
*hosts.**	Файл конфигурации локальной сети
hushlogins	Если существует файл file ~/.hushlogin или /etc/hushlogins, осуществляется "тихий" вход (это отключает проверку e-mail и вывод последнего времени входа и сообщения дня (Message of Day)). Если существует файл /var/log/lastlog, то выводится время последнего входа в систему

idmapd.conf	Конфигурация демона idmapd
idn.conf	Файл конфигурации для idnkit
idnalias.conf	Псевдонимы кодировок для idnkit
inputrc	Позволяет задавать обработку отображения символов в специальных ситуациях. Используется редко
issue	Приглашение, которое выводится при входе в систему (ссылка на /usr/lib/issue)
issue.net	Приглашение, которое выводится при сетевом входе в систему (ссылка на /usr/lib/issue.net)
krb5.conf	Параметры Kerberos
ld.so.cache	Хэш-версия файла ld.so.conf. Создается утилитой ldconfig
ld.so.conf	Настройка динамического связывания во время выполнения
lesskey	Задает параметры преобразования некоторых символов/клавиш, вы не будете редактировать этот файл
libao.conf	Конфигурация библиотеки libao (обычно здесь задается аудио-драйвер по умолчанию)
libaudit.conf	Конфигурация библиотеки libaudit
login.defs	Контрольные определения для пакета shadow. Например, здесь можно задать количество неудачных попыток входа в систему и многие другие параметры
logrotate.conf	Задает параметры ротации журналов
machine-id	Содержит идентификатор машины

mail.rc	Файл конфигурации программы mail
manpath.config	Этот файл используется пакетом man-db для настройки путей man и cat
mime.types	Содержит список MIME-типов и соответствующих им расширений файлов
mke2fs.conf	Файл конфигурации программы mke2fs
motd	Содержит сообщение дня (Message of the Day). В зависимости от настроек системы может выводиться при входе в систему
mtools.conf	Данный файл является частью пакета mtools.conf
netconfig	Файл конфигурации сети. Теперь используется только с кодом TI-RPC в библиотеке libtirc
netgroup	Содержит описание конфигурации сетевых групп
networks	Статическая информация о сетевых именах
nfsmount.conf	Файл конфигурации монтирования NFS
nscd.conf	Конфигурационный файл для nscd (Name Service Cache)
nsswitch.conf	Параметры NSS (Network Service Switch)
ntp.conf	Файл конфигурации сервера времени ntpd
os-release	Содержит информацию о релизе: номер версии, кодовое имя и т.д. (ссылка на /usr/lib/os-release)
passwd	База данных паролей

permissions	Этот файл используется программой chkstat и косвенно некоторыми RPM-скриптами для проверки или установки прав и режимов файлов и каталогов при установке
printcap	Этот файл автоматически генерируется cupsd, его не нужно редактировать. Все изменения, внесенные в этот файл, будут потеряны
profile	Не изменяйте этот файл во избежание потери изменений во время очередного обновления системы. Если вам нужно изменить этот файл, отредактируйте /etc/profile.local, чтобы установить ваши локальные настройки, например, глобальные псевдонимы, переменные VISUAL и EDITOR и т.д.
protocols	Список IP-протоколов
python3start	Startup-сценарий Python 3 для сохранения истории интерпретатора и автодополнения имен
pythonstart	То же, что и python3start, но для старых версий Python
raw	Определяет параметры привязки raw-устройств к блочным устройствам
rc.splash	Сценарий, определяющий внешний вид индикатора начальной загрузки
resolv.conf	Конфигурационный файл для системы разрешения имен
rpc	Список протоколов удаленного вызова процедур (RPC)
rsyncd.conf	Файл конфигурации для rsyncd
rsyncd.secrets	Пароли rsyncd
screenrc	Параметры программы screen (менеджер экрана с эмуляцией терминала VT100/ANSI)

securetty	Содержит имена устройств терминалов (tty), на которых разрешает вход в систему пользователю *root*
services	Список служб (сервисов)
shadow	Пароли из файла /etc/passwd физически хранятся в /etc/shadow. Поэтому фактически в /etc/passwd хранится список пользователей, а пароли этих пользователей находятся в /etc/shadow, доступ к которому ограничен
shells	Список установленных в системе интерпретаторов
slp.conf	Файл конфигурации OpenSLP SPI
smartd.conf	Файл конфигурации демона smartd
sudoers	Позволяет определить, кому можно использовать команду *sudo*
suspend.conf	Некоторые параметры питания
sysctl.conf	Файл конфигурации sysctl. Кроме этого файла sysctl также читает параметры из файлов /etc/sysctl.d/*.conf, /run/sysctl.d/*.conf и некоторых других
ttytype	Содержит список терминалов и определяет их тип
usb_modeswitch	Параметры для пакета usb_modeswitch
vconsole.conf	Конфигурационный файл для виртуальной консоли
virc	Файл конфигурации текстового редактора **vi**
wgetrc	Параметры программы wget
xattr.conf	Задает, как обработать расширенные атрибуты при копировании между файлами

12.12.3. Подкаталоги с конфигурационными файлами

Далее мы "пройдемся" по подкаталогам каталога /etc с целью выяснить, что находится в каждом из них (табл. 12.6). Содержимое вашего каталога /etc может отличаться в зависимости от дистрибутива и уже установленных программ. Вполне вероятно, что у вас не будет некоторых каталогов, представленных в таблице 12.6, но будут некоторые другие каталоги, назначение которых можно найти или в справочной системе Linux или в Интернете.

Таблица 12.6. Подкаталоги каталога /etc

Каталог	Описание
NetworkManager	Содержит файл конфигурации Network Manager и файлы конфигураций сетевых соединений в некоторых дистрибутивах
PackageKit	PackageKit – это открытый и свободный набор приложений для обеспечения высокоуровнего интерфейса для различных менеджеров пакетов. Этот каталог содержит файлы конфигурации PackageKit
X11	Параметры графического интерфейса X11 (X Window)
abrt	Конфигурация abrtd – демона автоматических отчетов о сбоях
alternatives	Каталог альтернатив по умолчанию. Файл является частью подсистемы update-alternatives, которая обслуживает символические ссылки, определяющие команды, файлы и каталоги, используемые по умолчанию
audit и audisp	В этих каталогах находятся конфигурационные файлы демона аудита – auditd и его диспетчера событий (audit event dispatcher). Основной конфигурационный файл – /etc/audit/auditd.conf. В нем задается поведение демона и некоторые настройки, например, расположение журнала по умолчанию /var/log/audit/audit.log
apt	Конфигурация менеджера пакетов **apt**

bash_comletion.d	Параметры автодополнения командной строки для оболочки **bash**
binfmt.d	Демон *binfmt.d* настраивает дополнительные двоичные файлы для выполнения во время загрузки. В этом каталоге находятся его конфигурационные файлы
cockpit	Конфигурация панели управления Cockpit
cifs-utils	Пакет cifs-utils содержит средство монтирования ресурсов общего доступа SMB/CIFS в Linux. В этом каталоге находятся конфигурационные файлы пакета cifs-utils
crond.d	Содержит файлы, которые загружаются в память одновременно с файлами из каталога /var/spool/cron. После этого демон *cron* загружает содержимое файла /etc/crontab и начинает его обработку
cron.daily, cron.hourly, cron.mounthly, cron.weekly	Содержат сценарии, которые будут выполнены демоном *cron*, соответственно, ежедневно, ежечасно, ежемесячно и еженедельно
cups	Содержит параметры конфигурации системы печати CUPS (Common Unix Printing System)
cupshelpers	В пакете python-cupshelpers содержатся модули Python, которые помогают создавать приложения и утилиты с использованием Python-интерфейса к CUPS. В этом каталоге находятся параметры этого пакета
crypto-policies	Конфигурация крипто-политик
dbus-1	Содержит файлы конфигурации демона dbus-daemon
default	Содержит некоторые параметры по умолчанию, например, параметры загрузчика GRUB

depmod.d	depmod – программа для создания файла modules.dep и map-файла. В этом каталоге находятся ее файлы конфигурации. Как администратор сервера, вы можете о них просто забыть, они вам не пригодятся
dhcp	Конфигурация DHCP-сервера
dnf	Конфигурация менеджера пакетов dnf
dracut.conf.d	dracut заменяет mkinitrd для создания загрузочной файловой системы в оперативной памяти (ramdisk). Здесь находятся конфигурационные файлы dracut
firewalld	Конфигурация брандмауэра
fonts	Содержит конфигурационные файлы подсистемы шрифтов. В частности, файл /etc/fonts/fonts.conf описывает каталоги со шрифтами, каталоги с кэшем шрифтов, а также описывает аналоги шрифтов
gnupg	Содержит конфигурационный файл GnuPg
grub.d	Содержит файлы, относящиеся к загрузчику GRUB. Вам не нужно редактировать эти файлы, они предназначены для служебных целей
init.d	Содержит сценарии системы инициализации
iproute2	Параметры подсистемы маршрутизации. Например, могут использоваться для IP-балансировки, то есть объединения нескольких Интернет-каналов в один
iscsi	Параметры iSCSI
java	Параметры Java

joe	Содержит конфигурационные файлы текстового редактора **joe**
jvm, jvm-common	Параметры виртуальной машины Java
ld.so.conf.d	Содержит файлы с расширением *conf*, которые используются для поиска разделяемых библиотек
libnl	Конфигурационные файлы библиотеки libnl
logrotate.d	Конфигурация средства ротации журналов
mc	Файлы конфигурации файлового менеджера Midnight Commander
mcelog	Программа mcelog позволяет расшифровать аппаратные ошибки. Настраивается посредством конфигурационных файлов в каталоге /etc/mcelog
modprobe.d	modprobe — программа для добавления и удаления модулей из ядра Linux. Соответственно в одноименном подкаталоге каталога /etc находятся ее конфигурационные файлы
modules-load.d	Содержит параметры некоторых модулей ядра. По умолчанию в этом каталоге пусто
openldap	Содержит конфигурационные файлы сервера каталогов OpenLDAP
opt	Файлы конфигурации для /opt/
pam.d	Параметры конфигурации модулей аутентификации PAM (Pluggable Authentication Modules)
pkcs11	Параметры программы pkcs11, используемой для управления объектами данных, которые находятся на зашифрованных устройствах PKCS#11 (Cryptoki)
pki	Содержит список GPG-ключей

plymouth	Plymouth — свободный графический экран загрузки для Linux. Этот каталог содержит его конфигурационные файлы
pm	Содержит конфигурационные файлы пакета pm-utils. Пакет pm-utils – это инфраструктура управления питанием нового поколения
postfix	Конфигурационные файлы почтового агента Postfix
ppp	Файлы конфигурации протокола PPP
pptp.d	Файлы конфигурации демона pptpd (протокол PPTP)
products.d	Относится к системе миграции между версиями дистрибутива openSUSE. Дополнительная информация может быть найдена по ссылке http://doc.opensuse.org/products/draft/SLES/SLES-deployment_sd_draft/cha.update.sle.html
pulse	PulseAudio – это звуковой сервер для POSIX-систем. Его основное назначение – смешивать звуковые потоки от разных приложений, что позволяет нескольким потокам воспроизводиться одновременно. Здесь находятся конфигурационные файлы PulseAudio
rc.d	Ссылка на каталог init.d
rsyslog.d	Конфигурация демона протоколирования rsyslogd
rpm	Различные параметры системы управления пакетами RPM. Обычно файлы из этого каталога не требуют изменения. Дополнительную информацию можно получить по адресу http://wiki.opennet.ru/RPM
samba	Конфигурационные файлы Samba

sasl2	Параметры SASL (Simple Authentication and Security Layer)
security	Еще один параметр конфигурации, относящийся к модулям аутентификации PAM
selinux	Содержит файлы конфигурации системы безопасности SELinux
skel	При создании новой учетной записи пользователя создается его домашний каталог в каталоге /home, при этом в созданный домашний каталог пользователя копируются файлы из каталога /skel. Все помещенные в этот каталог файлы будут скопированы в созданный домашний каталог
ssh	Содержит файлы конфигурации SSH-клиента и SSH-сервера
ssl	Файлы конфигурации OpenSSL
sudoers.d	Кроме файла /etc/sudoers, настройки sudo могут определяться содержимым файлов из каталога /etc/sudoers.d
sysconfig	Содержит конфигурационные файлы всей системы. В этом каталоге очень много различных конфигурационных файлов. Например, в каталоге /etc/sysconfig/network вы найдете конфигурационные файлы сетевых интерфейсов. А в файле **clock** хранится выбранный при установке часовой пояс
sysctl.d	sysctl.d настраивает параметры ядра при загрузке, здесь находятся его конфигурационные параметры
systemd	Конфигурационные файлы демона systemd
udev	Файлы конфигурации и файлы правил менеджера устройств udev

wpa_supplicant	Конфигурационные файлы пакета wpa_supplicant (обеспечивает поддержку WEP, WPA и WPA2)
xdg	xdg-open – это независимый пользовательский инструмент для настройки приложений рабочего стола по умолчанию. В этом каталоге находятся его конфигурационные файлы. Например, в каталоге /etc/xdg/autostart находятся все программы, которые могут быть запущены автоматически. Однако запускаются лишь те, которым разрешен запуск в определенной сессии
xinetd.d	Содержит дополнительные файлы конфигурации суперсервера xinetd (в современных дистрибутивах не используется)
xml	Конфигурационные файлы библиотеки libxml

12.13. Псевдофайловые системы

Псевдофайловые системы sysfs (каталог /sys) и proc (каталог /proc) используются для настройки системы и получения различной информации о системе и процессах. Свое название псевдофайловые системы получили из-за того, что они работают на уровне виртуальной файловой системы. В итоге оба эти средства (назовем их так) для конечных пользователей выглядят как обычная файловая система – вы можете зайти как в каталог /sys, так и в каталог /proc. В обоих этих каталогах будут файлы, вы можете просмотреть эти файлы и даже изменить их содержимое.

Содержимое многих файлов псевдофайловой системы /proc формируется "на лету". Обратите внимание на размер любого файла в каталоге /proc – он равен нулю, но если открыть файл, то информация в нем будет. Например, в файле /proc/version находится информация о версии Linux.

Монтирование файловых систем sysfs и proc осуществляется или в сценариях инициализации системы или через /etc/fstab. В последнем случае записи монтирования псевдофайловых систем выглядят так:

```
sysfs    /sys       sysfs   defaults        0 0
proc     /proc      proc    defaults        0 0
```

12.13.1. Псевдофайловая система *sysfs*

Файловая система sysfs (каталог /sys) предоставляет пользователю информацию о ядре Linux, об имеющихся в системе устройствах и драйверах этих устройств. На рис. 12.4 представлено содержание каталога /sys. В нем вы найдете следующие подкаталоги:

- **block** – содержит каталоги для всех блочных устройств, которые есть в вашей системе в настоящее время. Здесь под устройством подразумевается наличие физического устройства и его драйвера. Если вы подключите внешний жесткий диск, то в каталоге /sys/devices появится новое устройство, но в каталоге /sys/block оно появится только, если в системе есть драйвер для работы с этим устройством или же драйвер (модуль) встроен в само ядро
- **bus** – здесь находится список шин, которые поддерживает ваше ядро. Заглянув в этот каталог, вы обнаружите подкаталоги pci, pci_express, scsi и т.д. В каждом из этих каталогов будут подкаталоги devices и drivers. В первом находится информация об устройствах, подключенных к данной шине, во втором – информация о драйверах устройств
- **class** – позволяет понять, как устройства формируются в классы. Для каждого класса есть отдельный подкаталог в каталоге class
- **devices** – содержит дерево устройств ядра, точнее структуру файлов и каталогов, которая полностью соответствует внутреннему дереву устройств ядра
- **firmware** – содержит интерфейсы, предназначенные для просмотра и манипулирования firmware-специфичными объектами и их параметрами
- **fs** – информация о файловых системах, которые поддерживает ваше ядро
- **kernel** – общая информация о ядре
- **module** – здесь вы найдете подкаталоги для каждого загруженного модуля ядра. Имя подкаталога соответствует имени модуля. В каждом из под-

каталогов модулей вы найдете подкаталог parameters, содержащий специфичные для модуля параметры

* **power** – позволяет управлять параметрами питания, а также переводить систему из одного состояния питания в другое. Далее будет показано несколько примеров



Рис. 12.4. Содержание каталога /sys

Довольно интересен с практической точки зрения каталог /sys/power. В файле state находится состояние питание. Изменив должным образом содержимое этого файла, можно изменить состояние питания. Например, вот как можно перевести систему в состояние "Suspend to RAM", когда питание процессора отключается, но питание на память подается, благодаря чему ее содержимое не уничтожается:

```
$ sudo echo -n mem > /sys/power/state
```

При желании можно отправить систему в состояние "Suspend to Disk", когда содержимое памяти будет записано на жесткий диск, после чего питание будет отключено:

```
$ sudo echo -n disk > /sys/power/state
```

12.13.2. Псевдофайловая система *proc*

Файловая система **proc** позволяет отправлять информацию ядру, модулям и процессам. Вы можете не только получать информацию о процессах, но изменять параметры ядра и системы "на лету". Эта файловая система интересна тем, что позволяет изменять такие параметры ядра, которые невозможно изменить другим способом, к тому же вносимые изменения вступают в силу сразу же.

Некоторые файлы в /proc доступны только для чтения – вы можете только просмотреть их. А некоторые вы можете изменять, и эти изменения сразу же отразятся на работе системы. Просмотреть файлы из /proc можно любой программой для просмотра файлов, проще всего в консоли использовать команду *cat*:

```
cat /proc/<название файла>
```

Записать информацию в файл можно с помощью команды *echo*, как уже было показано выше:

```
sudo echo "информация" > /proc/<название файла>
```

В каталоге /proc очень много файлов и рассмотреть все мы не сможем. Прежде, чем мы приступим к самым интересным с моей точки зрения файлам, нужно понять, что означают каталоги с числами. Эти каталоги содержат информацию о запущенных процессах.

Итак, самые полезные информационные файлы:

- /proc/cmdline – содержит параметры, переданные ядру при загрузке
- /proc/cpuinfo – содержит информацию о процессоре, откройте этот файл, думаю, вам будет интересно. Кроме общей информации о процессоре вроде модели и частоты здесь выводится точная частота, размер кэша и псевдорейтинг производительности, выраженный в BogoMIPS. Значение BogoMIPS показывает "сколько миллионов раз в секунду компьютер может абсолютно ничего не делать". Способ измерения производительности пусть и не самый удачный, но от него до сих пор не отказались, а "на дворе" уже 3-я версия ядра
- /proc/devices – список устройств
- /proc/filesystems – полный список поддерживаемых вашим ядром файловых систем
- /proc/interrupts – информация по прерываниям
- /proc/ioports – информация о портах ввода/вывода

- /proc/meminfo – полная информация об использовании оперативной памяти. Как по мне, вывод этого файла более понятен и удобен, чем вывод команды *free*
- /proc/mounts – содержит список подмонтированных файловых систем
- /proc/modules – список загруженных модулей и их параметры
- /proc/swaps – содержит список активных разделов и файлов подкачки
- /proc/version – здесь находится версия ядра

Используя /proc можно не только получить информацию о системе, но и изменить ее. Например, в файлах /proc/sys/kernel/hostname и /proc/sys/kernel/domainname содержится информация об имени компьютера и домена. Вы можете не только просмотреть, но и изменить содержимое этих файлов, изменив, соответственно, имя узла и имя домена. Хотя практика изменения доменных имен через /proc/sys практикуется не часто, никто не мешает вам это сделать:

```
sudo echo "server" > /proc/sys/kernel/hostname
sudo echo "example.com" > /proc/sys/kernel/domainname
```

Файл /proc/sys/kernel/ctrl-alt-del позволяет регулировать тип перезагрузки системы при нажатии комбинации клавиш Ctrl + Alt + Del. По умолчанию в этом файле содержится значение 0, что означает так называемую "мягкую перезагрузку" (soft reboot). Если же вы внесете в этот файл значение 1, то при нажатии Ctrl + Alt + Del эффект будет такой же, как при нажатии кнопки Reset на корпусе компьютера:

```
sudo echo "1" > /proc/sys/kernel/ctrl-alt-del
```

Файл /proc/sys/kernel/printk позволяет задать, какие сообщения ядра будут выведены на консоль, а какие – попадут в журнал демона *syslog*. По умолчанию в этом файле содержатся значения 4 4 1 7. Сообщения с приоритетом 4 и ниже (первая четверка) будут выводиться на консоль. Вторая четверка – это уровень приоритета по умолчанию. Если для сообщения не задан уровень приоритета, то считается, что его приоритет будет равен 4.

Третье значение определяет номер самого максимального приоритета. Последнее значение – это уровень приоритета по умолчанию для первого значения.

В большинстве случаев изменяют только первое значение, позволяющее определить, будет ли сообщения с указанным уровнем приоритета выводиться на консоль или нет. Остальные параметры оставляют без изменения.

В файле /proc/sys/net/core/netdev_max_backlog содержится максимальное число пакетов в очереди. Значение по умолчанию – 1000.

Файл /proc/sys/fs/file-max позволяет изменить максимальное количество заголовков файлов, которое может быть одновременно открыто. Другими словами, этот файл задает, сколько одновременно может быть открыто файлов. Значение по умолчанию для ядра 3.16 и файловой системы btrfs – 73054.

Чтобы сохранить внесенные "на лету" изменения, и чтобы их не пришлось снова вводить при следующей перезагрузке сервера, нужно отредактировать файл /etc/sysctl.conf. Представим, что вы изменили значение из файла /proc/sys/fs/file-max. Тогда в файл /etc/sysctl.conf нужно добавить строку:

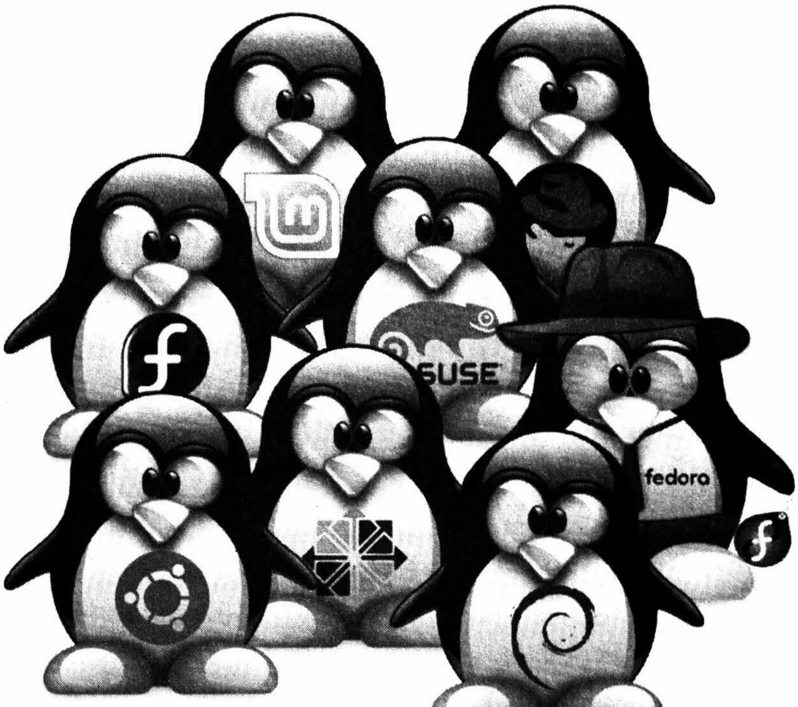
```
fs.file-max = 16 384
```

Принцип прост: /proc/sys/ отбрасывается совсем, а в оставшейся строке все слеши заменяются точками. Само же значение указывается через знак равенства. Если нужно указать несколько значений, то они указываются через пробел.

О файловой системе в Linux можно написать отдельную книгу, которая будет не меньше, чем та, которую вы держите в руках. Поэтому в этой главе мы рассмотрели только самое необходимое.

Глава 13.

Управление хранилищем



В этой главе мы рассмотрим довольно таки важные вещи – подключение нового жесткого диска и его разметка в классическом варианте – без всяких менеджеров томов, затем мы рассмотрим LVM и то, как можно расширить пространство группы томов, например, когда вы увеличили дисковое пространство виртуального сервера.

13.1. Подключение нового жесткого диска и его разметка

Классической программой для разметки жесткого диска в Linux и других операционных системах является программа **fdisk**. Конечно, в той же Windows программа **fdisk** совсем другая, но названия программ совпадают.

Рассмотрим пример использования этой программы. Представим, что мы подключили новый жесткий диск и нам нужно "ввести" его в эксплуатацию. Тренироваться лучше всего в виртуальной машине, особенно, если вы в первый раз осуществляете разметку диска.

Формат вызова **fdisk** такой:

```
# fdisk <устройство>
```

Да, команду **fdisk** нужно запускать с правами *root*. Далее мы будем считать, что новым является устройство /dev/sdb:

```
# fdisk /dev/sdb
```

Посмотрите на рис. 13.1. Я запустил программу **fdisk** для нового и неразмеченного жесткого диска. Программа сообщила мне, что:

1. Все изменения хранятся только в памяти и не переносятся на жесткий диск до тех пор, пока вы их не запишите.
2. Устройство не содержало таблицы разделов и была создана таблица разделов DOS (по умолчанию).

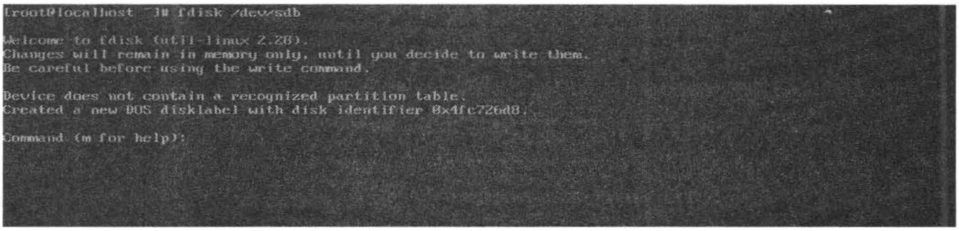


Рис. 13.1. Запуск fdisk для нового жесткого диска

Первым делом ознакомимся со списком команд `fdisk`. Введите команду *m* для получения справки. Список команд в последних версиях `fdisk` разбит на группы (рис. 13.2). В таблице 13.1 приведен список команд `fdisk`.

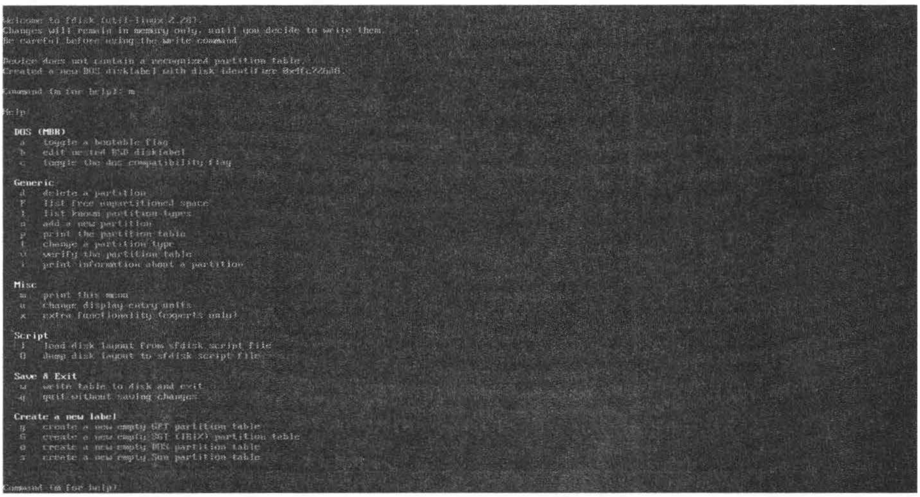


Рис. 13.2. Список команд fdisk

Таблица 13.1. Команды программы fdisk

Команда	Описание
a	Сделать раздел активным. Данный флаг был нужен для старых версий Windows, которые не могли загружаться с неактивных разделов. Сейчас эта команда попросту не нужна, а в мире Linux – тем более
b	Редактировать вложенную BSD-метку
c	Применить флаг совместимости с DOS
d	Удалить раздел
l	Вывести известные типы разделов
n	Добавить новый раздел
p	Вывод таблицы разделов
t	Изменить тип раздела
v	Проверить таблицу разделов
m	Вывод справки
u	Изменить единицы измерения
x	Дополнительная функциональность (только для экспертов)
w	Записать таблицу разделов на диск и выйти
q	Выход без сохранения изменении
g	Создать новую пустую таблицу разделов GPT
G	Создать новую пустую таблицу разделов SGI (для ОС IRIX)
o	Создать новую пустую таблицу разделов DOS
s	Создать новую пустую таблицу разделов Sun

Наша задача – создать раздел (или несколько разделов, здесь решать вам) и подмонтировать его (их) к корневой файловой системе.

Первым делом выведем таблицу разделов командой p (рис. 13.3). Как видно из рис. 13.3, таблица разделов пуста, а размер нашего жесткого диска всего 60 Гб.

```
Command (m for help): p
Disk /dev/sdb: 60 GiB, 64424509440 bytes, 125829120 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x4fc726d8

Command (m for help):
```

Рис. 13.3. Пустая таблица разделов

Наш диск довольно скромного размера, поэтому программа создала таблицу разделов DOS. Для больших жестких дисков лучше создать таблицу разделов GPT. Если программа неправильно выбрала тип таблицы разделов или вы хотите изменить его принудительно, введите команду g. Посмотрите на рис. 13.4: я изменил тип таблицы разделов, а затем опять отобразил таблицу разделов. Она по-прежнему пуста, но обратите внимание на ее тип – теперь у нас таблица разделов GPT.

> **Примечание.** Таблица разделов GPT (GUID Partition Table) является частью стандарта EFI (Extensible Firmware Interface) – стандарта, который был предложен компанией Intel на смену стандарта BIOS. Таблица GPT использует современную систему адресации логических блоков (LBA), а не старую систему CHS (цилиндр-головка-сектор). Но самое главное – это размер раздела. В GPT можно создать раздел размером до 9.4 Збайт (9.4 $\times 10^{21}$ байт), а в MBR – максимальный размер раздела всего 2.2 Тб (2.2 $\times$ 10^{21} байт).

```
Command (m for help): g
Created a new GPT disklabel (GUID: 299E41E1-4AB9-42E6-911F-C39BC06E7DE3).

Command (m for help): p
Disk /dev/sdb: 60 GiB, 64424509440 bytes, 125829120 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: gpt
Disk identifier: 299E41E1-4AB9-42E6-911F-C39BC06E7DE3

Command (m for help):
```

Рис. 13.4. Изменение таблицы разделов

Настало время создать раздел. Введите команду *n*. Программа попросит вас ввести (рис. 13.5):

- Номер раздела – это первый раздел, поэтому введите 1. В принципе, когда вы будете создавать второй раздел, программа автоматически предложит вам ввести номер 2.
- Первый сектор раздела. Просто нажмите **Enter** – программа автоматически предложит правильный вариант.
- Последний сектор раздела. Если вы хотите создать раздел на весь жесткий диск (то есть использовать все доступное пространство), тогда просто нажмите **Enter**. В противном случае укажите размер раздела. Проще всего это сделать, используя модификаторы +M и +G, например, для создания раздела размером 30 Гб укажите +30G. Если у вас очень большой жесткий диск, где пространство измеряется терабайтами, используйте модификатор T, например, +1T.

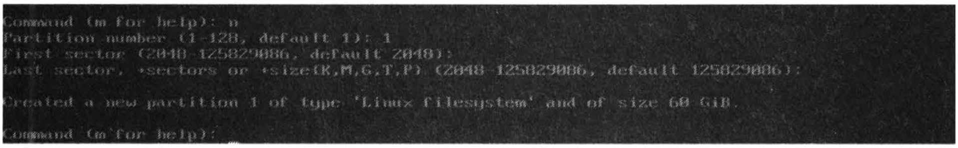


Рис. 13.5. Создание нового раздела

Программа сообщит вам, что один раздел создан. Также будет сообщен размер раздела. По умолчанию тип раздела – файловая система Linux (Linux filesystem). Если вы хотите изменить тип раздела, введите команду:

```
t <номер раздела>
```

Например:

```
t 1
```

Далее нужно или ввести код типа раздела или ввести команду *L* для вывода подсказки (рис. 13.6). Проблема вся в том, что нет способа постраничного просмотра типов разделов, а все они не помещаются на одном экране. Поэтому все равно придется обращаться к документации. Однако в большинстве случаев изменять тип раздела не нужно, поскольку при создании раздела он создается уже нужного типа. Исключение может возникнуть разве что для раздела подкачки (Linux swap). Его код – 82.

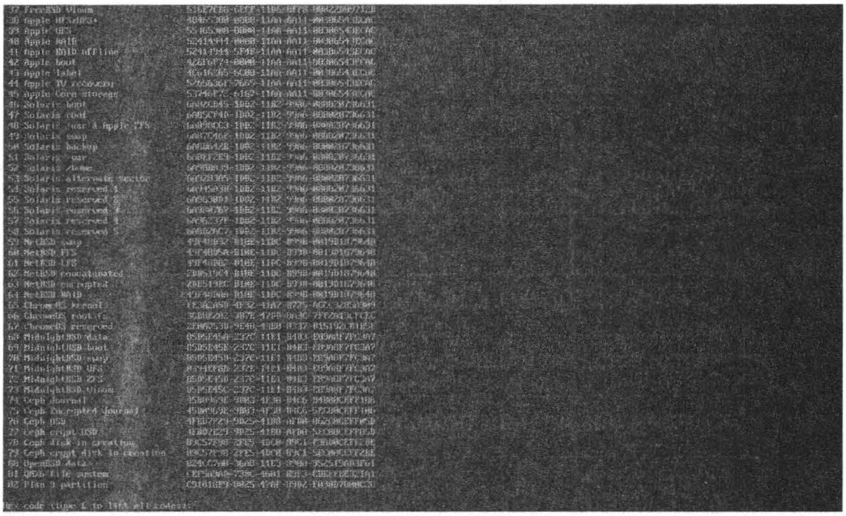


Рис. 13.6. Подсказка по типу раздела

Если вы передумали менять тип раздела, просто нажмите **Enter**. Теперь введите *p* для просмотра нашей таблицы разделов. Всегда просматривайте таблицу перед ее записью. После этого введите команду *w* для сохранения изменений и выхода из программы (рис. 13.7).

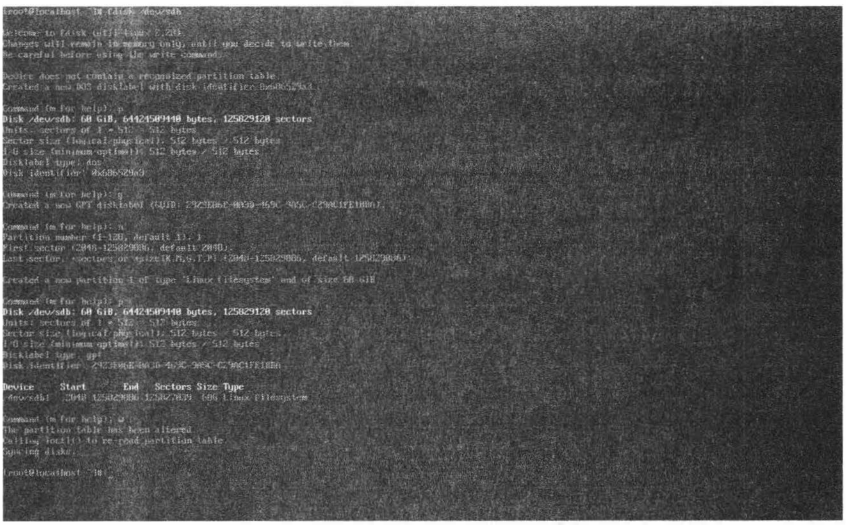


Рис. 13.7. Весь сеанс разметки диска: от создания таблицы разделов до записи изменений

Создать раздел мало. Нужно еще создать файловую систему. Не будем ничего выдумывать и создадим стандартную файловую систему ext4 командой mkfs.ext4:

```
# mkfs.ext4 /dev/sdb1
```

Результат выполнения этой команды приведен на рис. 13.8.

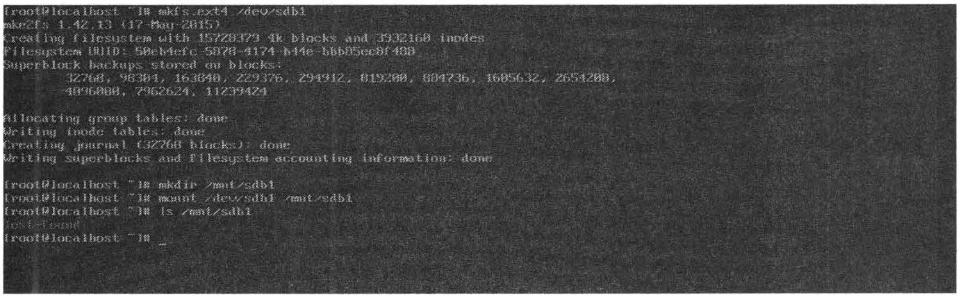


Рис. 13.8. Создание файловой системы и монтирование жесткого диска

После этого нужно создать точку монтирования для нового раздела и подмонтировать раздел (название точки монтирования можете изменить по своему усмотрению):

```
# mkdir /mnt/sdb1
# mount /dev/sdb1 /mnt/sdb1
# ls /mnt/sdb1
```

Почти все готово. Осталось только добавить запись в /etc/fstab для автоматического монтирования созданного раздела:

```
/dev/sdb1 /mnt/sdb1 ext4 defaults     1 1
```

Вот теперь можно приступать к использованию нового жесткого диска.

13.2. Менеджер логических томов

13.2.1. Введение в LVM

Впервые **менеджер логических томов** (LVM, Logical Volume Manager) появился в ядре 2.4 (его первая версия LVM 1), но более активно он стал применяться только в дистрибутивах с ядром 2.6 (уже вторая версия LVM 2).

Некоторые современные дистрибутивы используют LVM по умолчанию создают пулы LVM, в некоторых же LVM даже не установлен по умолчанию. Чтобы понять, нужен ли вам LVM, нужно разобраться, что это такое. В этой главе все будет изложено максимально доступно, так что бояться использовать LVM не стоит. В тоже время, если вам необходимы точные академические определения и дополнительная информация, обратитесь к Linux LVM Howto (*http://tldp.org/HOWTO/LVM-HOWTO/*), в котором много технических подробностей (если они вам нужны) и не совсем все сразу понятно.

LVM – это дополнительный уровень абстракции над аппаратными средствами, позволяющий собрать вместе несколько дисков в один логический диск, а затем разбить его так, как вам хочется.

Примеров использования LVM множество. Самый простой из них – объединение нескольких небольших дисков в один диск большего размера. Например, вам досталось даром (или почти даром) несколько SSD-дисков небольшого размера, скажем, по 128 Гб и вы хотите объединить эти 2-3 диска, чтобы получить один большой диск 256-384 Гб.

Второй пример тоже часто распространен. Представим, что система у вас установлена на небольшом диске, пусть даже на том же SSD-диске размером 80 Гб. Для Linux такой объем вполне достаточен, но рано или поздно свободное место закончится (все зависит от файлов, с которыми вы работаете). Вы покупаете диск большего размера, скажем, на 500 Гб или даже на 1 Тб. Но система уже установлена и переустанавливать ее не хочется. Что делать? Здесь вам поможет LVM.

Первый приведенный мною пример (объединение трех дисков во время установки) слишком тривиален и с ним справится любой, даже самый начинающий пользователь – просто во время установки нужно выбрать LVM (если, конечно, дистрибутив его поддерживает) и выбрать диски, которые вы объединяете в группу (пул).

Второй пример более сложный только за счет того, что система уже установлена, и мы договорились, что переустанавливать ее не будем. Поэтому он и заслуживает рассмотрения в этой главе, а дополнительную информацию вы сможете найти в Linux VVM Howto, ссылка на который была приведена ранее.

13.2.2. Уровни абстракции LVM

Прежде, чем перейти к рассмотрению практической стороны вопроса, нужно разобраться с тремя уровнями абстракции, с которыми вам придется столкнуться в LVM. Вот эти уровни:

1. PV (Physical Volume) — физические тома. Это могут быть разделы или целые, еще не размеченные диски.
2. VG (Volume Group) — группа томов. Физические тома объединяются в группу и создается единый диск, который вы можете разбить так, как вам хочется.
3. LV (Logical Volume) – логический раздел. Это раздел нашего единого диска (VG), который вы можете отформатировать в любую файловую систему и использовать так, как вам хочется, как обычный раздел обычного жесткого диска.

Прежде, чем мы продолжим, вы должны знать об одном недостатке LVM. Тома LVM не поддерживаются загрузчиком GRUB. Поэтому если вы используете этот устаревший загрузчик, вам нужно создать отдельный раздел /boot за пределами LVM. Грубо говоря, если у вас есть диск /dev/sda, раздел /dev/sda1 должен монтироваться к /boot. В него будет установлен загрузчик. Размер этого раздела должен быть небольшой, примерно 100 Мб (вполне будет достаточно).

Что же касается GRUB2, то он нормально загружается с LVM и никакой дополнительный раздел создавать не нужно.

13.2.3. Немного практики

Первым делом вам нужно установить пакет lvm2, если он еще не установлен.

```
# apt-get install lvm2
# yum install lvm2
```

Команда установки этого пакета для Fedora/CentOS приведена на всякий случай, так как в большинстве случаев этот пакет в этих дистрибутивах установлен по умолчанию.

Будем считать, что система сейчас установлена на /dev/sda1, который подмонтирован как /. Больше никаких разделов на этом диске не создано – для упрощения примера. Мы подключили второй жесткий диск, который пока еще не разбит. Имя этого диска – /dev/sdb.

Если на втором диске не созданы разделы, то создавать их и не нужно. Вы можете сдать все устройство сразу физическим томом (PV). Если на нем

есть разделы – не беда, вы можете добавить в группу томов все разделы поочередно.

Тратить время на создание разделов не хочется, тем более, что это и не нужно, поэтому создаем PV на все устройство /dev/sdb. Для этого используется команда *pvcreate*:

```
# pvcreate /dev/sdb
Physical volume "/dev/sdb" successfully created
```

Теперь нужно создать группу томов с помощью команды *vgcreate*. Данной команде нужно передать имя группы (пусть это будет my_vg) и указать физическое устройство:

```
# vgcreate my_vg /dev/sdb
Volume group "vg0" successfully created
```

Создаем отдельные логические тома (команда *lvcreate*) для раздела подкачки (swap) и разделов /home, /tmp и /var. Параметр *-L* задает размер раздела, например, -L30G задает размер 30 Гб. Параметр *-n* задает имя логического тома:

```
# lvcreate -n swap -L8G my_vg
# lvcreate -n home -L500G my_vg
# lvcreate -n var -L30G my_vg
# lvcreate -n tmp -L5G my_vg
```

Последний параметр – это имя нашей группы. При желании можно создать отдельный раздел и для /usr, но, как правило, программное обеспечение (а оно в основном устанавливается в /usr) в Linux много места не занимает. В общем, смотрите сами – никто не мешает ввести еще одну команду *lvcreate.* Тем более что у нас еще осталось место (если учитывать, что у нас жесткий диск на 1 ТБ).

Теперь разберемся, что и где мы создали. Просмотреть информацию по физическим томам, группам томов и логическим разделам можно с помощью команд *pvdisplay*, *vgdisplay* и *lvdisplay* соответственно.

Созданные нами разделы будут храниться в папке /dev/my_vg/. В этом каталоге вы найдете файлы home, tmp, var (правда, это будут ссылки, а не файлы, но суть от этого не меняется).

Когда созданы логические тома, можно создать на них файловые системы (отформатировать их). Вы можете использовать любую файловую систему, я предпочитаю ext4:

```
# mkfs.ext4 -L var /dev/my_vg/var
# mkfs.ext4 -L home /dev/my_vg/home
# mkfs.ext4 -L tmp /dev/my_vg/tmp
# mkswap -L swap /dev/my_vg/swap
# swapon /dev/my_vg/swap
```

Первые три команды создают файловую систему ext4 на устройствах /dev/my_vg/var, /dev/my_vg/home и /dev/my_vg/tmp. Последние две создают раздел подкачки и активируют его.

Настало время заняться перемещением данных. Суть в следующем – нужно подмонтировать поочередно новые тома и скопировать в них содержимое /home и /var:

```
# mkdir /mnt/home
# mkdir /mnt/var

# mount /dev/my_vg/home /mnt/home
# mount /dev/my_vg/var /mnt/var

# cp -a /home/* /mnt/home
# cp -a /var/* /mnt/var

# umount /mnt/home
# umount /mnt/var
```

В папку /tmp копировать ничего не нужно. Нужно только изменить права доступа:

```
# mkdir /mnt/tmp
# mount /dev/my_vg/tmp /mnt/tmp
# chmod -R a+rwx /mnt/tmp
# umount /tmp
```

Почти все. Теперь нужно добавить в /etc/fstab записи, монтирующие файловые системы /home, /var, /tmp и указать в нем раздел подкачки:

```
/dev/mapper/my_vg-home   /home  ext4 relatime        1 1
/dev/mapper/my_vg-home   /var        ext4 relatime   1 1
/dev/mapper/my_vg-tmp    /tmp        ext4 noatime    0 2
/dev/mapper/my_vg-swap   none        swap sw         0 0
```

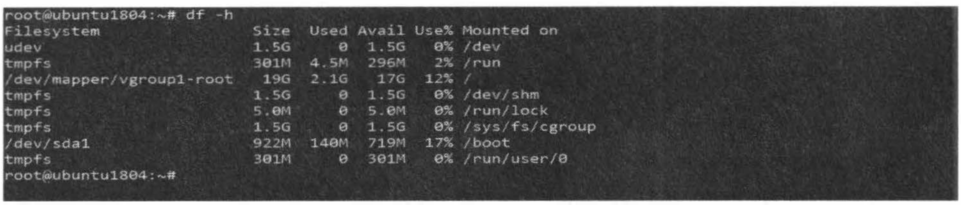
Все готово. Осталось только ввести команду *reboot*, чтобы система перезагрузилась. Корневая файловая система осталась на старом жестком диске (как и /usr), а каталоги, которые занимают больше всего места, были перемещены на логические разделы LVM.

13.3. Расширение LVM-пространства

Предположим, что вы расширили дисковое пространство виртуального сервера. Однако одного расширения в панели управления недостаточно – чтобы система увидела изменения, нужно произвести определенные действия. В принципе, задачу расширения пространства сервера можно было решить иначе, например, добавить еще один виртуальный жесткий диск, а дальше или использовать классический способ (раздел 13.1) или же добавить диск в группу томов (раздел 13.2) – все зависит от того, как настроен операционная система виртуального сервера. Но случилось то, что случилось – вы уже расширили жесткий диск, а вернуть ресурсы в пул, как правило, нельзя. Поэтому рассмотрим процедуру расширения виртуального диска.

Первым делом, посмотрим, сколько сейчас дискового пространства доступно. Для этого используется уже известная команда *df* с параметром *–h*, чтобы вывод был в удобочитаемом формате:

```
df -h
```



```
root@ubuntu1804:~# df -h
Filesystem                  Size  Used Avail Use% Mounted on
udev                        1.5G     0  1.5G   0% /dev
tmpfs                       301M  4.5M  296M   2% /run
/dev/mapper/vgroup1-root     19G  2.1G   17G  12% /
tmpfs                       1.5G     0  1.5G   0% /dev/shm
tmpfs                       5.0M     0  5.0M   0% /run/lock
tmpfs                       1.5G     0  1.5G   0% /sys/fs/cgroup
/dev/sda1                   922M  140M  719M  17% /boot
tmpfs                       301M     0  301M   0% /run/user/0
root@ubuntu1804:~#
```

Рис. 13.9. Команда df –h

На данный момент общий размер группы томов /dev/mapper/vgroup1-root составляет 19 Гб. Однако мы расширили диск и теперь нам нужно расширить размер этой группы томов до полного размера диска.

Чтобы система увидели новый объем жесткого диска, нужно пересканировать аппаратную конфигурацию. Для этого мы будем использовать следующую команду:

```
echo 1 > /sys/block/sda/device/rescan
```

Запустите утилиту **parted** (используется для работы с разделами диска):

```
parted
```

Введите команду *p* для просмотра имеющихся разделов (рис. 13.10). Запомните номер раздела, который мы будем расширять (2) и новый размер диска (42.9GB).

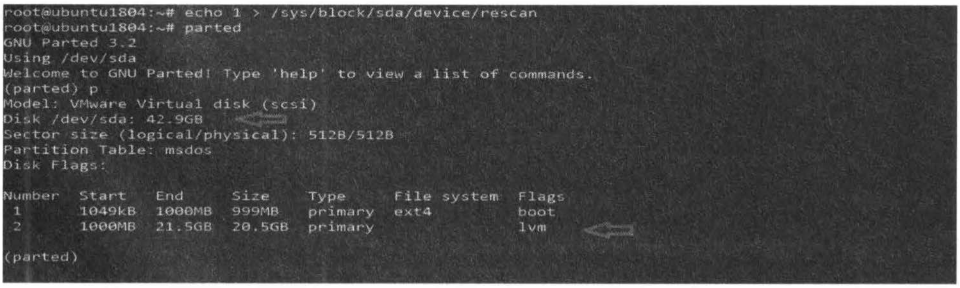


Рис. 13.10. Текущая таблица разделов

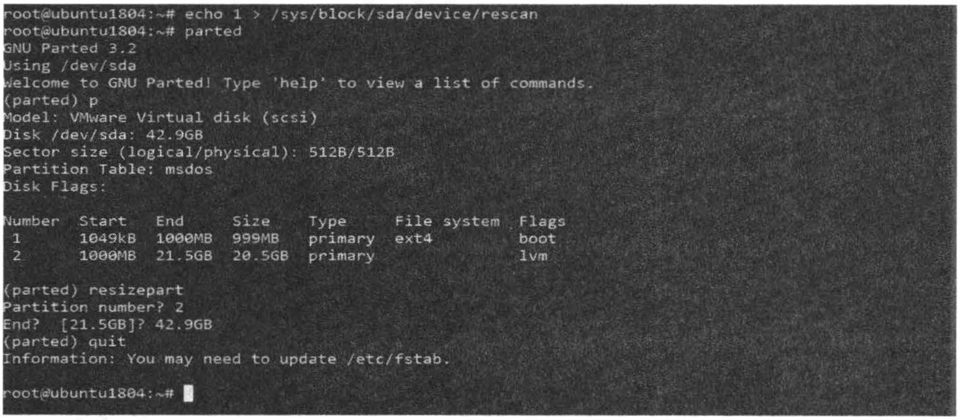
Запустим команду изменения раздела:

```
resizepart
```

Укажем номер раздела:

```
Partition number? 2
```

А затем нужно указать как раз то самое значение 42.9GB – именно так, без пробелов.



```
root@ubuntu1804:~# echo 1 > /sys/block/sda/device/rescan
root@ubuntu1804:~# parted
GNU Parted 3.2
Using /dev/sda
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted) p
Model: VMware Virtual disk (scsi)
Disk /dev/sda: 42.9GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start   End     Size    Type     File system  Flags
 1      1049kB  1000MB  999MB   primary  ext4         boot
 2      1000MB  21.5GB  20.5GB  primary               lvm

(parted) resizepart
Partition number? 2
End?  [21.5GB]? 42.9GB
(parted) quit
Information: You may need to update /etc/fstab.

root@ubuntu1804:~#
```

Рис. 13.11. Изменение размера раздела

Введите команду *quit* для выхода из **parted**. Parted сделал свою работу. Осталось сообщить ядру об изменениях размера:

```
pvresize /dev/sda2
  Physical volume "/dev/sda2" changed
  1 physical volume(s) resized / 0 physical volume(s) not resized
```

Расширим логический том:

```
lvextend -r -l +100%FREE /dev/mapper/vgroup1-root
```

По окончанию работ введем *df –h*, чтобы убедиться, что дисковое пространство расширилось.

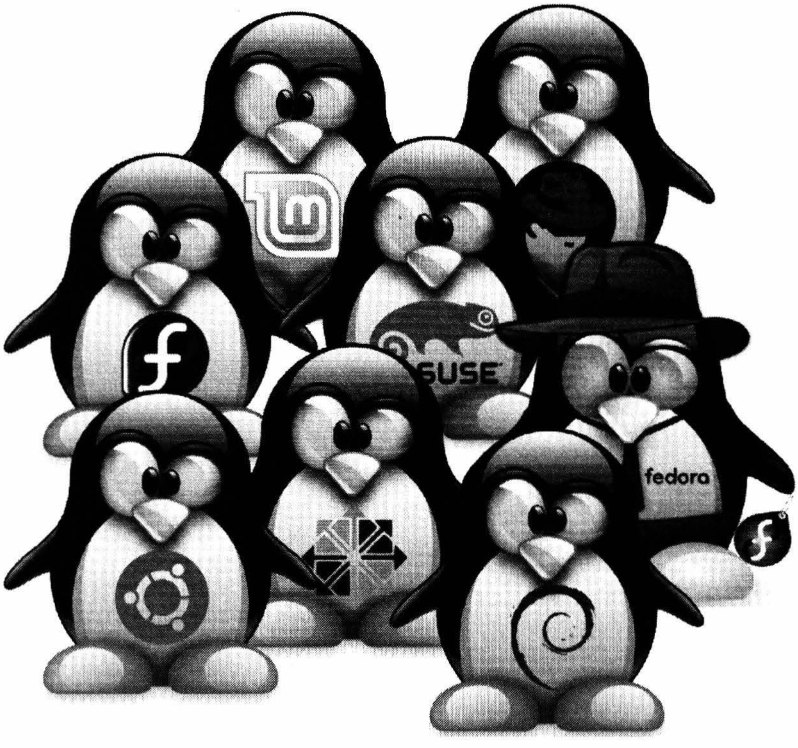
Посмотрите на рис. 13.12. На нем результат выполнения команд *pvresize*, *lvextend* и *df*. Последний вывод сообщает там, что размер группы томов vgroup1-root теперь составляет 41 Гб. Мы успешно расширили том до нового размера.

```
root@ubuntu1804:~# pvresize /dev/sda2
  Physical volume "/dev/sda2" changed
  1 physical volume(s) resized / 0 physical volume(s) not resized
root@ubuntu1804:~# lvextend -r -l +100%FREE /dev/mapper/vgroup1-root
  Size of logical volume vgroup1/root changed from 18.11 GiB (4637 extents) to <38.07 GiB (9745 exte
nts).
  Logical volume vgroup1/root successfully resized.
meta-data=/dev/mapper/vgroup1-root isize=512    agcount=12, agsize=400640 blks
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=1        finobt=1 spinodes=0 rmapbt=0
         =                       reflink=0
data     =                       bsize=4096   blocks=4748288, imaxpct=25
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0 ftype=1
log      =internal               bsize=4096   blocks=2560, version=2
         =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
data blocks changed from 4748288 to 9978880
root@ubuntu1804:~# df -H
Filesystem                Size  Used Avail Use% Mounted on
udev                      1.6G     0  1.6G   0% /dev
tmpfs                     315M  4.8M  311M   2% /run
/dev/mapper/vgroup1-root   41G  2.3G   39G   6% /
tmpfs                     1.6G     0  1.6G   0% /dev/shm
tmpfs                     5.3M     0  5.3M   0% /run/lock
tmpfs                     1.6G     0  1.6G   0% /sys/fs/cgroup
/dev/sda1                 967M  147M  754M  17% /boot
tmpfs                     315M     0  315M   0% /run/user/0
root@ubuntu1804:~#
```

Рис. 13.12. Том расширен

Глава 14.

Управление загрузкой Linux



14.1. Загрузчики Linux

Существует несколько загрузчиков Linux. На сегодняшний день основным загрузчиком является GRUB2, который устанавливается по умолчанию во всех современных дистрибутивах Linux.

Одним из самых "древних" загрузчиков является LILO (LInux LOader). Этот загрузчик давно уже не используется и ему на смену пришел загрузчик GRUB (GRand Unified Bootloader). GRUB является более гибким загрузчиком и "понимает" много разных файловых систем, в том числе FAT/FAT32, ext2, ext3, ReiserFS, XFS, BSDFS.

На смену GRUB пришел загрузчик GRUB2. Его отличия – очень запутанный и неудобный файл конфигурации, но время не стоит на месте, и если вы хотите использовать последние новинки в мире файловых систем Linux, а именно файловую систему ext4 и загрузку с LVM, вы должны использовать GRUB2. Также GRUB2 поддерживает UEFI, но это пригодится вам, только если вы – счастливый обладатель жесткого диска объема 2 Тб или больше.

Собственно, GRUB2 сейчас устанавливается во всех современных дистрибутивах, и нет смысла возвращаться на GRUB. Если возникнет необходимость, вы можете вернуться на обычный GRUB, установив пакет **grub-legacy**, но такая возможность есть только для платформы x86.

14.2. Загрузчик GRUB2

14.2.1. Конфигурационные файлы

В каталоге /etc/grub.d хранятся шаблоны, определяющие настройки GRUB2. Также некоторые его параметры хранятся в файле /etc/default/grub. По шаблонам из /etc/grub.d и файлу /etc/default/grub программой /usr/sbin/grub-mkconfig создается рабочий конфигурационный файл /boot/grub/grub.cfg, который по задумке разработчиков GRUB2 вы не должны редактировать вручную.

Поэтому есть две стратегии настройки GRUB2. Первая заключается в непосредственном редактировании файла /boot/grub/grub.cfg. Загрузчику GRUB2 все равно, кто или что отредактирует этот файл – или вы или программа *grub-mkconfig*. Вторая заключается в редактировании файлов из каталога /etc/grub.d и файла /etc/default/grub. После чего вы будете должны ввести команду *grub-mkconfig* для создания файла /boot/grub/grub.cfg по заданным вами настройкам.

Чтобы решить, какая из стратегий для вас лучше, нужно знать формат и содержимое всех этих файлов. Начнем с основного файла конфигурации, который сложнее и длиннее файла конфигурации обычного GRUB (см. лист. 14.1).

Листинг 14.1. Файл конфигурации /boot/grub/grub.cfg

```
#
# Не редактируйте этот файл вручную!
#
# Он автоматически генерируется программой grub-mkconfig по шаблонам
# из /etc/grub.d и настройкам из /etc/default/grub
#

### НАЧАЛО файла /etc/grub.d/00_header ###
if [ -s $prefix/grubenv ]; then
  load_env
fi
# Загрузочная метка по умолчанию
set default="0"
if [ "${prev_saved_entry}" ]; then
  set saved_entry="${prev_saved_entry}"
  save_env saved_entry
  set prev_saved_entry=
  save_env prev_saved_entry
```

```
  set boot_once=true
fi

function savedefault {
  if [ -z "${boot_once}" ]; then
    saved_entry="${chosen}"
    save_env saved_entry
  fi
}

function load_video {
  insmod vbe
  insmod vga
  insmod video_bochs
  insmod video_cirrus
}

insmod part_msdos
insmod ext2
# Корневое устройство
set root='(hd0,msdosl)'
search --no-floppy --fs-uuid --set=root b7300e54-fff5-4f31-8002-bea43c64f344
if loadfont /usr/share/grub/unicode.pf2 ; then
  set gfxmode=640x480
  load_video
  insmod gfxterm
  insmod part_msdos
  insmod ext2
  set root='(hd0,msdosl)'
  search --no-floppy --fs-uuid --set=root b7300e54-fff5-4f31-8002-bea43c64f344
  set locale_dir=($root)/boot/grub/locale
  set lang=ru_RU
  insmod gettext
fi
terminal_output gfxterm
set timeout=5
### КОНЕЦ файла /etc/grub.d/00_header ###

### НАЧАЛО файла /etc/grub.d/05_debian_theme ###
insmod part_msdos
insmod ext2
# Корневое устройство
set root='(hd0,msdosl)'
search --no-floppy --fs-uuid --set=root b7300e54-fff5-4f31-8002-bea43c64f344
insmod png
if background_image /usr/share/images/desktop-base/joy-grub.png;
then
  set color_normal=white/black
  set color_highlight=black/white
```

```
else
  set menu_color_normal=cyan/blue
  set menu_color_highlight=white/blue
fi
### КОНЕЦ файла /etc/grub.d/05_debian_theme ###

### НАЧАЛО файла /etc/grub.d/10_linux ###
# Содержит главную загрузочную метку. Далее мы ее рассмотрим подробнее
menuentry 'Debian GNU/Linux, Linux 3.2.0-4-amd64' --class debian --class gnu-
linux --class gnu --class os {
   load_video
   insmod gzio
   insmod part_msdos
   insmod ext2
   set root='(hd0,msdos1)'
   search --no-floppy --fs-uuid --set=root b7300e54-fff5-4f31-8002-
bea43c64f344
   echo 'Загружается Linux 4.2.0-4-amd64 ...'
   linux /boot/vmlinuz-4.2.0-4-amd64 root=UUID=b7300e54-fff5-4f31-8002-
bea43c64f344 ro initrd=/install/gtk/initrd.gz quiet
   echo 'Загружается начальный ramdisk ...'
   initrd  /boot/initrd.img-4.2.0-4-amd64
}
menuentry 'Debian GNU/Linux, Linux 4.2.0-4-amd64 (recovery mode)' --class
debian --class gnu-linux --class gnu --class os {
   load_video
   insmod gzio
   insmod part_msdos
   insmod ext2
   set root='(hd0,msdos1)'
   search --no-floppy --fs-uuid --set=root b7300e54-fff5-4f31-8002-
bea43c64f344
   echo 'Загружается Linux 4.2.0-4-amd64 ...'
   linux /boot/vmlinuz-4.2.0-4-amd64 root=UUID=b7300e54-fff5-4f31-8002-
bea43c64f344 ro single initrd=/install/gtk/initrd.gz
   echo 'Загружается начальный ramdisk ...'
   initrd  /boot/initrd.img-4.2.0-4-amd64
}
### КОНЕЦ /etc/grub.d/10_linux ###

### НАЧАЛО /etc/grub.d/20_linux_xen ###
### КОНЕЦ /etc/grub.d/20_linux_xen ###

### НАЧАЛО /etc/grub.d/20_memtest86+ ###
# Метка для memtest86 - программы для проверки памяти
menuentry "Memory test (memtest86+)" {
   insmod part_msdos
   insmod ext2
   set root='(hd0,msdos1)'
```

```
   search --no-floppy --fs-uuid --set=root b7300e54-fff5-4f31-8002-
bea43c64f344
   linux16 /boot/memtest86+.bin
}
menuentry "Memory test (memtest86+, serial console 115200)" {
   insmod part_msdos
   insmod ext2
   set root='(hd0,msdos1)'
   search --no-floppy --fs-uuid --set=root b7300e54-fff5-4f31-8002-
bea43c64f344
   linux16 /boot/memtest86+.bin console=ttyS0,115200n8
}
menuentry "Memory test (memtest86+, experimental multiboot)" {
   insmod part_msdos
   insmod ext2
   set root='(hd0,msdos1)'
   search --no-floppy --fs-uuid --set=root b7300e54-fff5-4f31-8002-
bea43c64f344
   multiboot  /boot/memtest86+_multiboot.bin
}
menuentry "Memory test (memtest86+, serial console 115200,
experimental multiboot)" {
   insmod part_msdos
   insmod ext2
   set root='(hd0,msdos1)'
   search --no-floppy --fs-uuid --set=root b7300e54-fff5-4f31-8002-
bea43c64f344
   multiboot  /boot/memtest86+_multiboot.bin console=ttyS0,115200n8
}
### КОНЕЦ /etc/grub.d/20_memtest86+ ###

# Далее этот файл я немного сократил, поскольку дальше в нем нет
ничего интересного
```

Файл огромный и его синтаксис напоминает синтаксис bash-сценариев. Если вы просмотрели этот конфигурационный файл, то вы уже догадались, что делает программа grub-mkconfig: она собирает воедино все файлы из каталога /etc/grub.d (кстати, в листинге 14.1 перечислена большая часть из этих файлов) и вносит в общий конфигурационный файл из /etc/default/grub.

Основная запись из всего листинга 14.1 – это запись *menuentry*. Именно в таких записях описываются элементы меню загрузчика GRUB.

```
menuentry 'Debian GNU/Linux, Linux 4.2.0-4-amd64' --class debian --class gnu-
linux --class gnu --class os {
   load_video
```

```
   insmod gzio
   insmod part_msdos
   insmod ext2
   set root='(hd0,msdos1)'
   search --no-floppy --fs-uuid --set=root b7300e54-fff5-4f31-8002-
bea43c64f344
   echo 'Loading Linux 4.2.0-4-amd64 ...'
   linux /boot/vmlinuz-4.2.0-4-amd64 root=UUID=b7300e54-fff5-4f31-8002-
bea43c64f344 ro initrd=/install/gtk/initrd.gz quiet
   echo 'Loading initial ramdisk ...'
   initrd  /boot/initrd.img-4.2.0-4-amd64
}
```

В одинарных кавычках после *menuentry* указывается название загрузочной метки. Далее идут параметры, которые вообще можно не указывать и от этого Debian загружаться не перестанет. В фигурных скобках – основная конфигурация. Директива *load_video* – это ни что иное, как вызов функции *load_video*, которая также описана в этом файле конфигурации. Функция вставляет некоторые модули (команда *insmod*), необходимые для работы графического режима. Обратите внимание, что команды *insmod* загружают не модули ядра Linux, а модули GRUB2, которые находятся в каталоге /boot/grub.

Внутри {} можно использовать команду *echo* для обозначения различных этапов загрузки, что и сделано в нашем примере. Можете отказаться от *echo*, на загрузку это никак не повлияет.

Основные команды – это *linux* и *initrd*. Первая указывает путь к ядру Linux и задает параметры ядра. В нашем случае параметр ядра *root* указывает устройство, на котором находится корневая файловая система. Устройство указано в виде UUID. UUID-имена очень удобны. Представим, что у вас есть один жесткий диск SATA и два контроллера. Если вы подключите его ко второму контроллеру, обычное имя изменится (например, было /dev/sda, а стало /dev/sdb), а UUID-имя – нет. При желании, вы можете указать имя в старом формате, например, root=/dev/sda1. Параметр ядра *root* задает монтирование корневой файловой системы в режиме "только чтение" (это нормально, позже она будет перемонтирована), *initrd* – задает файл RamDisk, а последний параметр ядра **quiet** задает "тихую" загрузку ядра, при которой будут выводиться только самые важные сообщения.

Команда *initrd* задает путь к файлу *initrd*.

Теперь рассмотрим файл /etc/default/grub (листинг 14.2)

Листинг 14.2. Файл /etc/default/grub

```
# После редактирования этого файла запустите команду 'update-grub' для
# обновления файла /boot/grub/grub.cfg.
# Для получения полной информации об этом файле введите команду
#   info -f grub -n 'Simple configuration'

# Загрузочный элемент (menuentry по умолчанию)
GRUB_DEFAULT=0
# Таймаут
GRUB_TIMEOUT=5
# Задает название дистрибутива, не изменяйте эту строку
GRUB_DISTRIBUTOR=`lsb_release -i -s 2> /dev/null || echo Debian`
# Параметры ядра Linux по умолчанию
GRUB_CMDLINE_LINUX_DEFAULT="quiet"
# Еще одна строка для задания параметров ядра
GRUB_CMDLINE_LINUX="initrd=/install/gtk/initrd.gz"

# Раскомментируйте эту строку, если вы хотите отключить графический режим
#GRUB_TERMINAL=console

# Разрешение в графическом режиме
# Вы можете использовать только те режимы, которые ваша видеокарта
# поддерживает через VBE. Просмотреть список
# таких режимов можно с помощью команды `vbeinfo'
#GRUB_GFXMODE=640x480

# Расскомментируйте эту строку, если вы не хотите
# использовать UUID-имена устройств
#GRUB_DISABLE_LINUX_UUID=true

# Расскомментируйте, если хотите запретить генерирование меток восстановления
#GRUB_DISABLE_RECOVERY="true"

# Расскомментируйте, если хотите получить гудок при загрузке GRUB
#GRUB_INIT_TUNE="480 440 1"
```

Как видите, параметры из файла /etc/default/grub понятны и не нуждаются в особых комментариях. Но в нем мы узнали о еще одной команде – *update-grub*. Так какую из них использовать – *update-grub* или *grub-mkconfig*?

На самом деле это почти одна и та же команда. Дело в том, что команда *grub-mkconfig* по умолчанию выводит конфигурацию GRUB2 на экран, поэтому, чтобы она записалась в файла /boot/grub/grub.cfg, запускать ее нужно так:

```
sudo grub-mkconfig > /boot/grub/grub.cfg
```

Или же вы можете ввести команду *update-grub*, которая сделает то же самое. Другими словами, команда *update-grub* – это сценарий, который вызывает только что приведенную команду. Как по мне, то использовать команду *update-grub* удобнее.

Так какую стратегию GRUB2 использовать? Редактировать шаблоны и параметры или сразу конфигурационный файл? Если вы работаете за компьютером в гордом одиночестве и нет и не предвидится других администраторов, тогда можете выбрать ту стратегию, которая вам больше нравится.

Если же есть или планируются другие администраторы, то нужно редактировать шаблоны и параметры вместо редактирования конфигурационного файла вручную. Дело в том, что если вы внесете изменения непосредственно в конфигурационный файл, а потом другой администратор захочет изменить какой-то незначительный параметр, например, добавить гудок при загрузке GRUB2, то команда *update-grub* перезапишет все сделанные вами изменения.

14.2.2. Выбор метки по умолчанию

Как правило, даже если у вас установлена одна только Linux, у вас будет несколько загрузочных меток (несколько записей *menuentry*). Выбрать метку по умолчанию можно с помощью параметра GRUB_DEFAULT. Нумерация меток начинается с 0, то есть первой метке соответствует значение 0.

После того, как вы установите другой номер метки по умолчанию нужно ввести команду *update-grub* и перезагрузить систему.

Другими словами, последовательность такая: редактируем файл /etc/default/grub, изменяем значение параметра GRUB_DEFAULT и вводим команду *update-grub*.

14.2.3. Загрузка Windows

Упор в этой книге делается на серверы и прочее производственное применение Linux, однако вдруг вы захотите установить Linux на домашний компьютер, скорее всего, ей придется "сожительствовать" с Windows.

Чтобы добавить возможность загрузки Windows из GRUB2, нужно отредактировать файл /etc/grub.d/40_custom и добавить в него следующие строки:

```
menuentry "Windows (on /dev/sda1)" {
        insmod part_msdos
        insmod ntfs
        set root='(hd0,msdos1)'
        search --no-floppy --fs-uuid --set UUID
        drivemap -s (hd0) ${root}
        chainloader +1
}
```

Здесь Windows находится на /dev/sda1 и адрес этого раздела указывается в *set root*. Значение UUID нужно заменить на UUID вашего Windows-раздела. Остальные параметры можете оставить без изменения (еще не забудьте исправить hd0 в drivemap, если Windows установлена не на первом жестком диске).

14.2.4. Пароль загрузчика GRUB2

Загрузчик GRUB позволял только установить пароль – или общий или на загрузку определенной метки. Загрузчик GRUB2 более гибкий в этом плане, поскольку вы можете настроить не только пароли, но и логины. Также есть минимальная система разграничения прав доступа.

Итак, в GRUB2 есть суперпользователь, который может редактировать загрузочные метки. Существует возможность восстановить пароль *root* путем передачи ядру параметра *init*. Но для этого нужно отредактировать конфигурацию GRUB2. Если вы установите пароль суперпользователя, то изменить конфигурацию загрузчика вы сможете только после ввода этого пароля.

Также в GRUB2 есть обычные пользователи, которые имеют право только выбирать загрузочную метку. Они не имеют права редактировать конфигурацию загрузчика. В принципе, можно обойтись одним паролем суперпользователя, но при желании GRUB2 может довольно гибко разграничить права пользователей.

Давайте сначала добавим пароль суперпользователя. Для этого в файл /etc/grub.d/00_header добавьте строки:

```
set superusers="main_admin"
password main_admin 123456789
```

Первая команда задает суперпользователя *main_admin*, а вторая – задает для него пароль. Старайтесь избегать общепринятых имен вроде *admin*, *root* и т.д. Так у злоумышленника, который хочет изменить конфигурацию GRUB2 будет две неизвестных.

Пароль пока в незашифрованном виде и это не очень хорошо. Поскольку если загрузиться с LiveCD или LiveUSB, то его можно будет увидеть. Позже я покажу, как зашифровать пароль.

Обычные пользователи задаются инструкцией *password*, например:

```
password me 12345
```

По сути *main_admin* – тоже был бы обычным пользователем, если бы не инструкция *set superusers*, которая делает его суперпользователем.

Представим, что у нас есть следующие строки:

```
set superusers="main_admin"
password main_admin 123456789
password me 12345
```

Пользователь *main_admin* может загружать операционные системы и редактировать конфигурацию GRUB2. Пользователь *me* может только загружать операционные системы.

Если вы хотите, чтобы определенные метки могли загружать только определенные пользователи, добавьте к *menuentry* параметр *--users*:

```
menuentry "Windows" --users me {
        insmod part_msdos
        insmod ntfs
        set root='(hd0,msdos1)'
        search --no-floppy --fs-uuid --set UUID
        drivemap -s (hd0) ${root}
        chainloader +1
}
```

Теперь зашифруем пароль. Введите команду:

```
grub-mkpasswd-pbkdf2
```

Программа запросит пароль, зашифрует его и выведет на экран его кэш. Вы увидите что-то подобное:

```
grub.pbkdf2.sha512.10000.9290F727ED06C38BA4549EF7DE25CF5642659
211B7FC076F2D27080136.887CFF169EA83D5235D8004742AA7D6187A41E31
87DF0CE14E256D85ED97A979080136.887CFF169EA8335235D8004242AA7D6
187A41E3187DF0CE14E256D85ED97A97357AAA8FF0A3871AB9EEFF458392F4
62F495487387F685B7472FC6C29E293F0A0
```

Данный хэш нужно указать вместо пароля пользователя. Однако вместо инструкции *passwd* нужно использовать *password_pbkdf2*. Например:

```
password_pbkdf2 me grub.pbkdf2.sha512.10000.9290F727ED06C38BA4
549EF7DE25CF5642659211B7FC076F2D27080136.887CFF169EA83D5235D80
04742AA7D6187A41E3187DF0CE14E256D85ED97A979080136.887CFF169EA8
335235D8004242AA7D6187A41E3187DF0CE14E256D85ED97A97357AAA8FF0A
3871AB9EEFF458392F462F495487387F685B7472FC6C29E293F0A0
```

После изменения файлов из каталога /etc/grub.d не забудьте ввести команду *upgrade-grub* для обновления основного файла конфигурации.

14.2.5. Установка загрузчика

Команда установки загрузчика такая же, как и в случае с GRUB:

```
# /sbin/grub-install <устройство>
```

Например:

```
# /sbin/grub-install /dev/sda
```

Поскольку GRUB2 у вас уже установлен, вряд ли вам придется вводить эту команду. Исключение может поставить разве что переустановка Windows, которая перезапишет загрузочный сектор своим загрузчиком. Поэтому вам

придется загрузиться с LiveCD, выполнить *chroot* для вашей старой корневой системы и ввести команду *grub-install* для установки загрузчика GRUB2.

14.3. Система инициализации

После своей загрузки ядро передает управление системе инициализации. Цель этой системы – выполнить дальнейшую инициализацию системы. Самая главная задача системы инициализации – запуск и управление системными службами.

Служба (сервис, демон) – специальная программа, выполняющаяся в фоновом режиме и предоставляющая определенные услуги (или, как говорят, сервис – отсюда и второе название). Что превращает обычный компьютер, скажем, в FTP-сервер? Правильно, запущенная служба FTP – тот же ProFTPD или подобная. Вы можете установить программу ProFTPD и настроить ее на автоматический запуск системой инициализации. Тогда при каждой загрузке наш компьютер будет превращаться в FTP-сервер. Аналогично и с другими сервисами – достаточно установить определенную программу, чтобы превратить компьютер в веб-сервер или почтовый сервер. Но стоит вам отключить ее и компьютер уже прекращает предоставлять обеспечиваемые программой услуги, следовательно, превращается в самый обычный компьютер.

В мире Linux существовало очень много разных систем инициализации – init, upstart, init-ng. Все их рассматривать уже нет смысла, поскольку в современных дистрибутивах используется современная система инициализации systemd.

systemd — подсистема инициализации и управления службами в Linux, фактически вытеснившая в 2010-е годы традиционную подсистему **init**. Основная особенность — интенсивное распараллеливание запуска служб в процессе загрузки системы, что позволяет существенно ускорить запуск операционной системы. Основная единица управления — модуль, одним из типов модулей являются «службы» — аналог демонов — наборы процессов, запускаемые и управляемые средствами подсистемы и изолируемые контрольными группами.

14.3.1. Принцип работы

Система инициализации **systemd** используется во многих современных дистрибутивах, в частности в Fedora, Ubuntu, CentOS и openSUSE. На данный момент – это самая быстрая система инициализации.

Давайте подумаем, как можно ускорить запуск Linux? Можно пойти по пути **upstart** – параллельно запускать службы. Но параллельный запуск – не всегда хорошо. Нужно учитывать зависимости служб. Например, сервис d-bus нужен многим другим сервисам. Пока сервис d-bus не будет запущен, нельзя запускать сервисы, которые от него зависят.

Если сначала запускать основные сервисы и ждать, пока они будут запущены, а потом уже запускать службы, которые от них зависимы, особого выигрыша в производительности по сравнению с **init** вы не увидите. Но если сервис d-bus (или любой другой, от которого зависят какие-то другие сервисы) запускается долго, то все остальные службы будут ждать его.

Как обойти это ограничение? При своем запуске службы проверяют, запущена ли необходимая им служба, по наличию файла сокета. Например, в случае с **d-bus** – это файл /var/run/dbus/system_bus_socket. Если мы создадим сокеты для всех служб, то мы можем запускать их параллельно, особо не беспокоясь, что произойдет сбой какой-то службы при запуске из-за отсутствия службы, от которой они зависят. Даже если несколько служб, которым нужен сервис d-bus, запустятся раньше, чем сам сервис d-bus: ничего страшного. Каждая из этих служб отправит в сокет (главное, что он уже открыт!) сообщение, которое обработает сервис d-bus после того, как он запустится. Вот и все.

Но это не единственное «ухищрение», посредством которого осуществляется ускорение запуска компьютера, инициализацию которого производит **systemd**. Эта система инициализации запускает только необходимые сервисы. Остальные же будут запущены по мере необходимости. Концепция отложенного запуска используется и в других операционных системах – например, в Mac OS X (там система инициализации называется launchd) и в Windows (концепция отложенного запуска служб). Так что решение не очень новое, но зато проверенное.

Основными функциями **systemd** являются:

- Активация на основании сокетов – система инициализации systemd прослушивает сокеты всех системных служб. Сокеты передаются системным службам сразу после запуска сервисов. Благодаря этому осуществляется параллельный запуск сервисов. Также это позволяет перезапускать сервисы без потери любых отправленных им сообщений, то есть пока сервис перезапускается, отправленные ему сообщения накапливаются, и он сможет их обработать после того, как будет запущен.

- Активация на основании устройств – systemd может запустить определенные службы, когда станет доступным определенный тип оборудования. Например, вы подключили Bluetooth-адаптер, может быть запущен сервис bluetooth.
- Активация на основании d-bus – служба инициализации может запустить сервисы, которые используют d-bus для межпроцессного взаимодействия, например, когда клиентское приложение попытается связаться с системной службой.
- Активация на основании путей – systemd может запустить службу, если изменится содержание каталога.
- Управление точками монтирования и автоматическим монтированием – система инициализации отслеживает и управляет точками монтирования и автоматического монтирования.
- Снимки системных состояний – благодаря этой возможности systemd может сохранить состояние всех модулей и восстановить предыдущее состояние системы.
- Параллелизация – systemd запускает системные службы параллельно благодаря активации на основании сокетов. Параллельная активация существенно сокращает время загрузки системы.
- Обратная совместимость с SysV – поддерживаются сценарии инициализации SysV, что упрощает переход на systemd. Однако все устанавливаемые в современных дистрибутивах пакеты служб уже адаптированы под systemd, поэтому не нужно надеяться, что во время установки пакета какого-то сервиса будут установлены SysV-сценарии. Будут созданы файлы, необходимые для запуска сервиса посредством systemd.

14.3.2. Конфигурационные файлы systemd

Обилие различных конфигурационных файлов systemd может ввести в ступор даже бывалого линуксоида, не говоря уже о пользователе, который впервые видит **systemd**. Когда я впервые познакомился с systemd, у меня было только одно желание – снести ее и установить вместо нее **init**. Но мы это не будем делать. Чтобы разобраться со всеми файлами, нужно понимать, как работает эта система.

В systemd используется концепция модулей (юнитов). Существующие типы модулей описаны в таблице 14.1.

Таблица 14.1. Типы модулей системы инициализации systemd

Тип	Описание
service	Служба (сервис, демон), которую нужно запустить. Пример имени модуля: network.service. Изначально systemd поддерживала сценарии SysV (чтобы управлять сервисами можно было, как при использовании init), но в последнее время в каталоге /etc/init.d систем, которые используют systemd практически пусто (или вообще пусто), а управление сервисами осуществляется только посредством systemd
target	Цель. Используется для группировки модулей других типов. В systemd нет уровней запуска, вместо них используются цели. Например, цель multi-user.target описывает, какие модули должны быть запущены в многопользовательском режиме
snapshot	Снимок. Используется для сохранения состояния systemd. Снимки могут использоваться для перевода системы из одного состояния в другое, например, в состояние сна и пробуждение
mount	Точка монтирования. Представляет точку монтирования. Система инициализации systemd контролирует все точки монтирования. При использовании systemd файл /etc/fstab уже не главный, хотя все еще может использоваться для определения точек монтирования
automount	Автоматическая точка монтирования. Используется для монтирования сменных носителей – флешек, внешних жестких дисков, оптических дисков и т.д.
socket	Сокет. Представляет сокет, находящийся в файловой системе или в Интернете. Поддерживаются сокеты AF_INET, AF_INET6, AF_UNIX. Реализация довольно интересная. Например, если сервису service1.servive соответствует сокет service1.socket, то при попытке установки соединения с service1.socket будет запущен service1.servive

device	Устройство. Представляет устройство в дереве устройств. Работает вместе с udev: если устройство описано в виде правила udev, то его можно представить в systemd в виде модуля device
path	Файл или каталог, созданный где-то в файловой системе
scope	Процесс, который создан извне
slice	Управляет системными процессами. Представляет собой группу иерархически организованных модулей
swap	Представляет область подкачки (раздел подкачки) или файл подкачки (свопа)
timer	Представляет собой таймер системы инициализации systemd

Модули хранятся в следующих каталогах:

- /etc/systemd/system/ – обладает самым высоким приоритетом. Здесь содержатся модули, которые созданы и управляются системным администратором.
- /run/systemd/system/ – модули, созданные во время выполнения. Приоритет этого каталога ниже, чем каталога /etc/systemd/system/, но выше, чем у /usr/lib/systemd/system.
- /usr/lib/systemd/system/ – модули, которые установлены из пакетов.

Типичный файл модуля типа service приведен в листинге 14.3.

Листинг 14.3. Типичный файл модуля типа *service*

```
[Unit]
Description=Daemon to detect crashing apps
After=syslog.target

[Service]
ExecStart=/usr/sbin/abrtd
Type=forking
```

```
[Install]
WantedBy=multi-user.target
```

В секции Unit содержится общая информация о сервисе. Эта секция есть и в других модулях, а не только в сервисах.

Секция **Service** содержит информацию о сервисе. Параметр **ExecStart** описывает команду, которую нужно запустить. Параметр **Type** указывает, как сервис будет уведомлять systemd об окончании запуска.

Секция **Install** содержит информацию о цели, в которой должен запускаться сервис. В нашем видно, что сервис будет запущен при активации цели multi–user.target.

Вы можете использовать эту «болванку» для написания собственного сервиса, который потом нужно поместить в файл /etc/systemd/system/имя_сервиса.service. После этого нужно перезапустить саму systemd, чтобы она узнала о новом сервисе:

```
# systemctl daemon-reload
```

14.3.3. Цели

Теперь поговорим о целях. Файлы целей *.target предназначены для группировки вместе других юнитов systemd через цепочку зависимостей. Так, модуль цели graphical.target, который используется для запуска графического сеанса, запускает системные службы GDM (файл gdm.service) и Accounts Service (accounts–daemon.service), а также активирует цель multi-user.target. В свою очередь, цель multi–user.target запускает другие системные службы, например, D-Bus (dbus.service) и активирует другие цели вроде basic.target.

В systemd имеются предопределенные цели, которые напоминают стандартный набор уровней запуска.

Некоторые цели называются runlevelN.target, чтобы упростить переход бывших пользователей **init** на **systemd**, а именно:

- poweroff.target (runlevel0.target) – завершение работы и отключение системы;
- rescue.target (runlevel1.target) – однопользовательский режим, среда восстановления;

- multi–user.target (runlevel2.target, runlevel3.target, runlevel4.target) – многопользовательский режим, без графического интерфейса;
- graphical.target (runlevel5.target) – многопользовательский режим с графическим интерфейсом
- reboot.target (runlevel6.target) – завершение работы и перезагрузка системы

Управление службами осуществляется с помощью программы systemctl. Подробнее о службах мы поговорим в следующем разделе, а пока разберемся, как использовать systemctl для завершения работы системы:

- systemctl halt – останавливает систему;
- systemctl poweroff – выключает систему;
- systemctl reboot – перезагружает систему.

Многим пользователям будет удобнее использовать старые команды *halt*, *poweroff* и *reboot*. Но все же теперь вы знаете, что есть альтернативные способы завершения работы.

14.4. Управление сервисами при использовании systemd

При использовании системы инициализации systemd управление службами осуществляется посредством программы systemctl. Команда *systemctl* используется для разных целей, поэтому в таблице 14.2 представлены не все ее параметры, а только те, которые имеют отношение к сервисам.

Таблица 14.2. Параметры программы systemctl

Параметр	Описание
start <имя.service>	Запускает сервис
stop <имя.service>	Останавливает сервис
restart <имя.service>	Перезапускает сервис

try-restart <имя.service>	Перезапуск сервиса только, если он запущен
reload <имя.service>	Перезагружает конфигурацию сервиса
status <имя.service>	Отображает подробное состояние сервиса
is-active <имя.service>	Отображает только строку active (сервис запущен) или inactive (остановлен)
list-units --type service --all	Выводит состояние всех сервисов
enable <имя.service>	Включает сервис (обеспечивает его автоматический запуск)
disable <имя.service>	Отключает сервис (сервис не будет автоматически запускаться при запуске системы)
reenable <имя.service>	Деактивирует сервис и сразу его использует
list-unit-files --type service	Выводит список всех сервисов и сообщает, какие из них активированы, а какие – нет

Примеры:

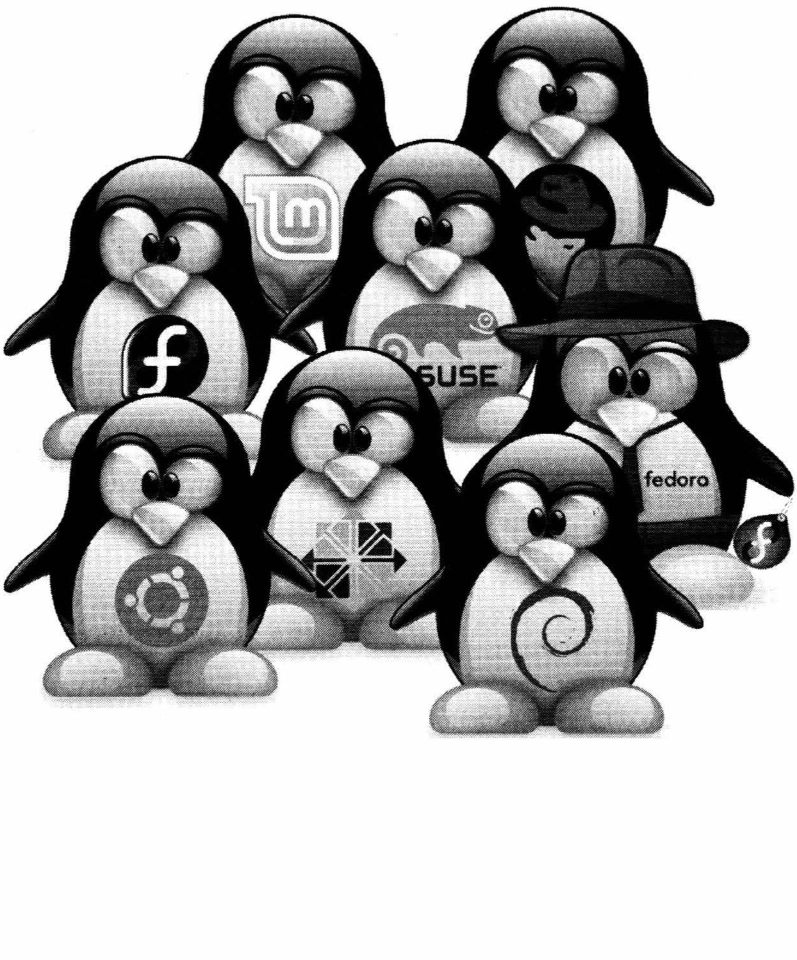
```
# systemctl start httpd.service
# systemctl stop httpd
```

Первая команда запускает сервис httpd (веб-сервер), вторая – останавливает. Обратите внимание, что «.service» можно не указывать.

Бывалые пользователи Linux сразу заметят удобства. Ранее, чтобы отключить службу на определенном уровне запуска, нужно было удалить ее символическую ссылку из определенного каталога. Аналогично, чтобы служба запускалась на определенном уровне запуска (например, в графическом режиме), нужно было создать символическую ссылку. Сейчас всего этого нет, а есть только команды *enable* и *disable*, что гораздо удобнее.

Глава 15.

Управление процессами



15.1. Команды *ps*, *nice* и *kill*

15.1.1. Получение информации о процессе

Современные операционные системы устроены так, что каждому процессу присваивается уникальный номер – PID (Process ID, ИД процесса), используя который можно управлять процессом, например, можно завершить процесс или изменить его приоритет.

Узнать PID можно с помощью команды `ps`. Команда `ps`, введенная без параметров, просто показывает список процессов, запущенных на текущем терминале. Видно, что сейчас запущен `bash` и сама команда `ps` (правда, на момент завершения вывода процесс с ID 975 уже не будет существовать, но на момент самого вывода такой процесс существовал), см. рис. 15.1.

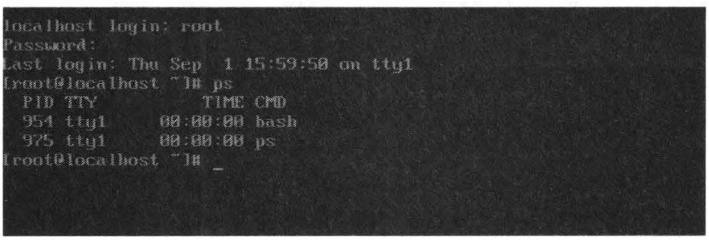


***Рис. 15.1. Команда* `ps`**

Параметр *-a* позволяет вывести список всех процессов пользователя. Посмотрите на рис. 15.2 на консоли `tty3` запущена программа `nano`, на консоли `tty1` – программа `ps`.

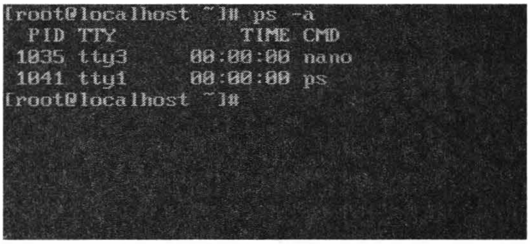


***Рис. 15.2. Команда* `ps -a`**

Если нужно вывести процессы какого-то определенного пользователя, тогда используйте параметр *-u*:

```
$ ps -u root
```

Вывести абсолютно все процессы можно с помощью опции *-A*. Обратите внимание на регистр опции! Поскольку в системе процессов будет очень много, лучше перенаправить вывод программы на команду `less` для более удобного просмотра (рис. 15.3):

```
$ ps -A | less
```

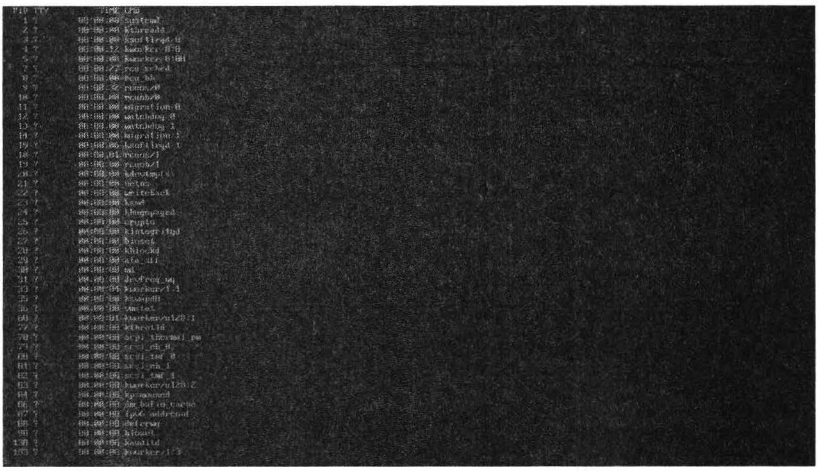


Рис. 15.3а. Команда* `ps -A | less` *(начало вывода)

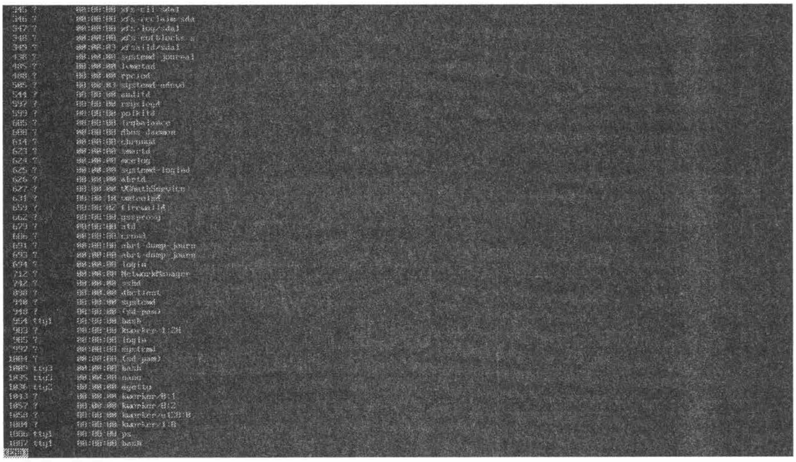


Рис. 15.3б. Команда **`ps -A | less`** *(завершение вывода)*

> **Примечание**. Для выхода из программы `less` нажмите q на клавиатуре. Листать вывод можно стрелками вверх и вниз.

Команда `ps` сортирует процессы по `PID`. Колонка `TTY` – это терминал, к которому привязан процесс. Если в этой колонке вы видите знак *?*, значит, процесс не привязан ни к одному из терминалов. Как правило, это системные процессы-службы. Они запускаются без привязки к терминалу. Чтобы отобразить только процессы без привязки к терминалу, используется опция *-x* (рис. 15.4).

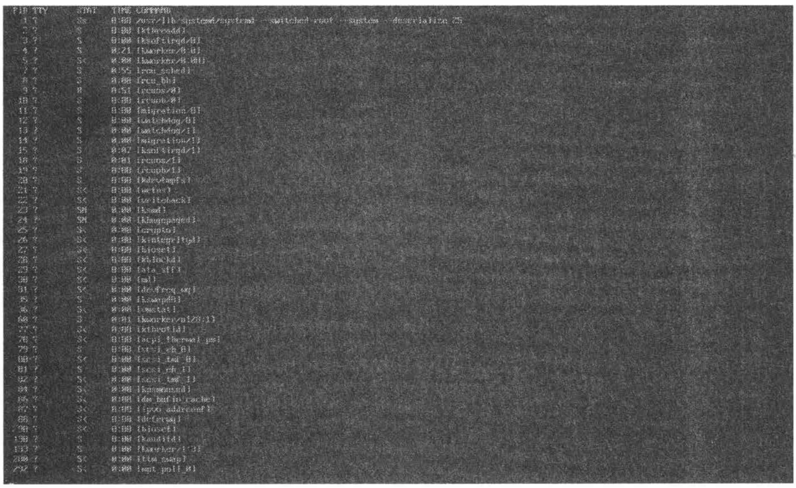


Рис. 15.4. Команда **`ps -x | less`**

Колонка `STAT` – это состояние, в котором находится процесс. Возможные значения для этой колонки приведены в таблице 15.1. Обратите внимание: колонка `STAT` есть только, когда программа запущена с параметром *-x*. Если программа запущена с другими параметрами, например, *-A*, вместо нее будет колонка `TIME`, которая сообщает занимаемое процессом процессорное время.

Таблица 15.1. Возможные состояния процесса

Состояние	Описание
D	Процесс в непрерывном сне (как правило, ожидает ввода/вывода)
R	Выполняется в данный момент
S	Ожидание (то есть процесс "спит" менее 20 секунд, после чего он переходит или в состояние R или в состояние D)
T	Процесс остановлен
t	То же, что и T, но причина остановки – остановка отладчиком
W	Процесс в свопинге (подкачке)
X	Процесс мертв, вы его никогда не увидите
Z	Процесс-зомби – он уже завершен, но не "похоронен" его родителем, то есть процесс-родитель еще не считал код завершения

У команды `ps` есть несколько синтаксисов установки параметров. Мы использовали BSD-синтаксис. Например, для вывода всех процессов в стандартном синтаксисе используется команда *-e*, а в нашем случае – *-A*. Вы вольны использовать любой синтаксис, но в случае с BSD-синтаксисом программа `ps` выводит дополнительное состояние процесса (работает подобно программе `ps` в системе BSD). Дополнительное состояние процесса описано в таблице 15.2.

Таблица 15.2. Дополнительное состояние процесса (BSD-синтаксис)

Состояние	Описание
<	Высокий приоритет
N	Низкий приоритет
L	У процесса есть страницы, заблокированные в памяти
s	Это лидер сессии
l	Процесс является многопотоковым
+	Находится на первом плане в группе процессов

Последняя колонка вывода `ps` – это `CMD`. Она содержит команду, которой был запущен процесс. Не просто название исполнимого файла, но путь (если он был указан в команде) и переданные программе параметры.

Если программа была запущена без указания полного пути к исполнимому файлу, и вы хотите знать, где он находится, введите команду `which`, например:

```
$ which nano
/bin/nano
```

Если вам нужно узнать PID определенного процесса, но вам не хочется просматривать длинный список системных процессов, используйте команду `grep`, как фильтр. Например, следующая команда позволит нам узнать PID процесса *sshd* (это SSH-сервер):

```
# ps -A | grep sshd
```

Если такой процесс не запущен, вывод будет пуст. Или же вы получите вывод вроде этого:

```
929 ?   00:00:00 sshd
```

15.1.2. Изменение приоритета процесса

Когда мы знаем PID процесса, мы можем изменить его приоритет. В некоторых случаях полезно изменить приоритет процесса. Например, можно повысить приоритет процесса, выполняющего резервное копирование, чтобы программа успела за ночь создать все необходимые резервные копии, и чтобы этот процесс утром потом не мешал нормальной работе сервера.

Запустить программу с определенным приоритетом можно командой `nice`:

```
# nice -n <приоритет> команда аргументы
```

Здесь приоритет задается от -20 (максимальный приоритет) до 19 (минимальный). Если процесс уже был запущен, и вы не можете его прерывать, но повысить приоритет нужно, используйте команду renice:

```
# renice -n <приоритет> -p PID
```

15.1.3. Аварийное завершение процесса

Если процесс завис и его нельзя завершить, как обычно, тогда для его аварийного завершения используется команда `kill`. Формат вызова этой команды следующий:

```
$ kill [опции] PID
```

Конечно, перед этим нужно узнать PID процесса. На рис. 15.5 изображена команда `kill` в действии: сначала я вывел список процессов, чтобы узнать PID процесса *nano* (1191), затем я ввел команду `kill 1191`, чтобы "убить" этот процесс. Наконец, я вывел список процессов еще раз, чтобы убедиться, что процесс *nano* завершен.

Используя параметры программы, можно по-разному завершить процесс. Самый *эффективный сигнал* *9* (KILL) – означает аварийное завершение процесса. Программа не может игнорировать или как-либо обработать этот процесс.

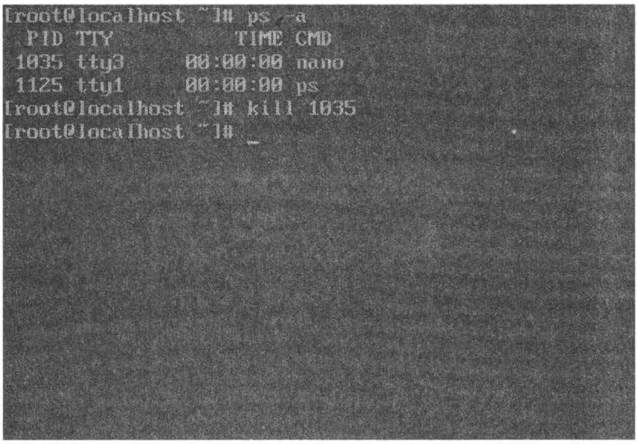


***Рис. 15.5. Использование команды* `kill`**

Если нужно попытаться корректно завершить работу программы, ей отправляют *сигнал 15* (TERM), означающий, что программа должна освободить все занятые ресурсы, сохранить все данные. Вот только если программа зависла и не отвечает на запросы пользователя, этот сигнал мало чем поможет, но попытаться стоит.

Сигнал 19 (STOP) позволяет временно приостановить работу программы, а *сигнал 18* (CONT) – возобновить приостановленный ранее процесс.

Для сетевых служб полезен *сигнал 1* (HUP), означающий, что процесс должен перезапуститься и перечитать файл конфигурации. Полезно, когда вы изменили файл конфигурации и хотите, чтобы демон был перезапущен (хотя для этого правильнее использовать команду `service`). Обычная программа при получении *сигнала 1* завершает работу.

Пример отправки сигнала:

```
$ kill -9 1035
```

Если вам лень получать PID процесса, можно завершить его и по имени, используя команду `killall`, например:

```
$ killall nano
```

Вот только если в вашей системе есть два процесса с именем *nano*, например, один на консоли `tty2`, а другой – на *tty4*, то будут завершены оба процесса. Если это то, что вам нужно, используйте `killall`, в противном случае лучше использовать команду `kill` для завершения именно того процесса, который можно завершить.

Еще есть команда `xkill`, позволяющая "убить" программу, имеющую графический интерфейс. Такие программы можно завершить и командой *kill*, но программа **xkill** предоставляет графический метод завершения. После ввода этой команды указатель мыши примет вид черепа. Для завершения программы нужно щелкнуть по ее окну.

15.2. Команда *top*

Как было отмечено ранее, программа `ps` по умолчанию сортирует процессы по колонке PID, а не по колонке `TIME`. Конечно, можно использовать различные параметры программы, чтобы добиться нужного нам вывода, но все равно программа не будет показывать ситуацию в реальном времени. Если же вам нужно знать, что происходит с вашими процессами в реальном времени, вам нужно использовать программу `top` (рис. 15.6).

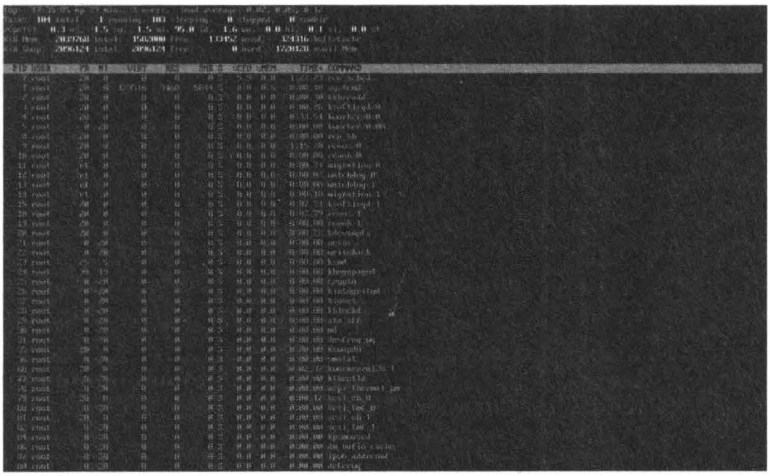


***Рис. 15.6. Команда* `top`**

Назначение колонок программы описано в таблице 15.3.

Таблица 15.3. Колонки программы top

Колонка	Описание
PID	PID процесса
USER	Владелец процесса (пользователь, запустивший программу)
PR	Приоритет процесса
NI	Значение nice (см. ранее)
VIRT	Виртуальная память, которая используется процессом
RES	Размер процесса, который не перемещается в область подкачки
SHR	Разделяемая память, используемая процессом
S	Состояние процесса (см. табл. 15.1)
%CPU	Процессорное время, занимаемое процессом в данный момент
%MEM	Память, используемая процессом
TIME+	Процессорное время, которое было потрачено с момента запуска процесса
COMMAND	Команда запуска процесса

При просмотре списка программы `top` вы можете управлять сортировкой процессов с помощью нажатия клавиши **F**, которая изменяет колонку, по которой сортируется список процессов (рис. 15.7). По умолчанию сортировка выполняется по колонке `%CPU`.

Нажатие клавиши <U> показывает только процессы определенного пользователя. После нажатия <U> нужно будет ввести имя пользователя, процессы которого вы хотите просмотреть, или нажать **Enter**, чтобы просмотреть процессы всех пользователей.

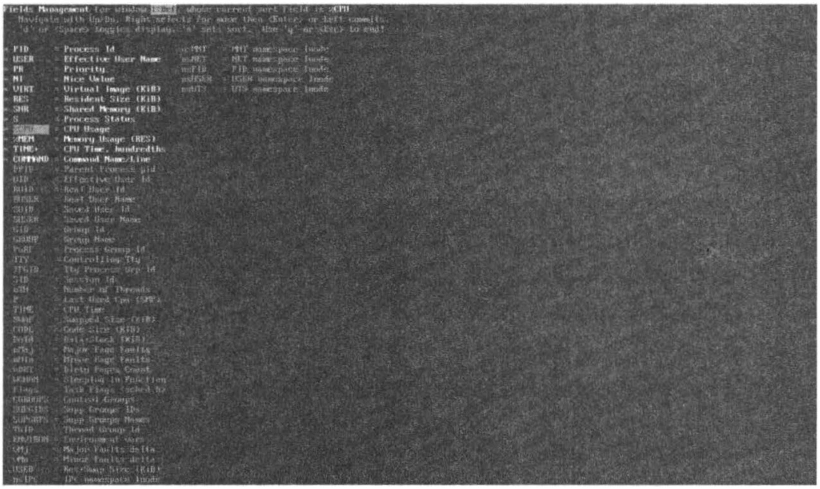


Рис. 15.7. Выбор колонки, по которой осуществляется сортировка

15.3. Информация об использовании памяти и дискового пространства

Хотя управление памятью и дисковым пространством не совсем относится к управлению процессами, но эти самые процессы активно "поедают", как память, так и дисковое пространство, поэтому иногда полезно знать, как просмотреть информацию об использовании памяти (команда `free`) и дискового пространства (команда `df`), см. рис. 15.8.

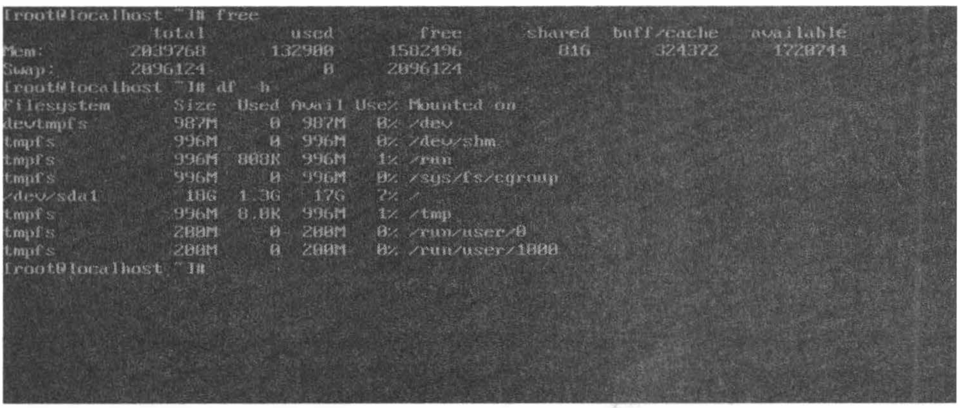


Рис. 15.8. Команды **`free`** *и* **`df -H`**

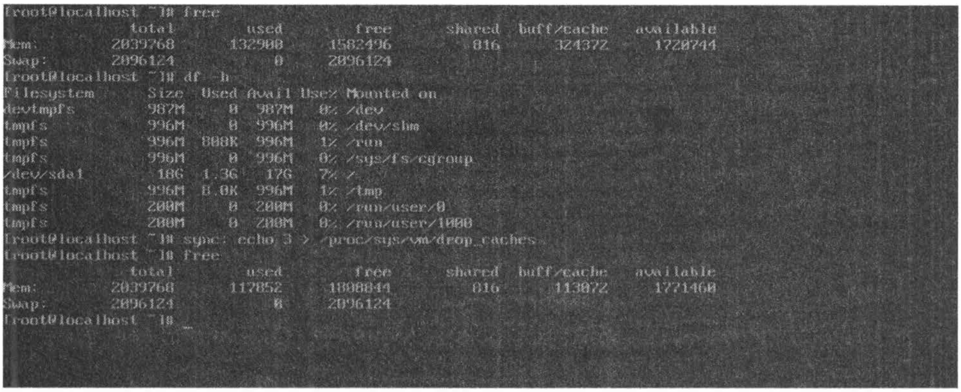
Программа free выводит информацию об использовании памяти (Mem) и подкачки (Swap). Колонка total – это общее количество памяти в килобайтах, used – использованное количество памяти (тоже в килобайтах), free – свободно памяти, shared – разделяемая память, buff/cache – размер кэша, available – общий объем доступной памяти.

Параметр *-H* команды df означает вывод информации об объеме в удобных для восприятия человеком единицах, то есть в мегабайтах и гигабайтах.

Обратите внимание на значение buff/cache в выводе команды free. Оно показывает сколько памяти задействовано под буфер ввода/вывода и кэш. В нашем случае (рис. 15.8) – примерно 323 Мб. На реальном сервере это значение будет гораздо выше. Немного освободить память можно, очистив кэш. Для этого введите команду:

```
sync; echo 3 > /proc/sys/vm/drop_caches
```

Сначала мы командой sync сбрасываем содержимое буферов на диск, а затем уничтожаем кэш. Если просмотреть затем информацию об использовании памяти, то вы увидите, что размер кэша был уменьшен почти в три раза (рис. 15.9). Однако помните, что эта команда может негативно отразиться на стабильности системы и на скорости ее работы. Не всегда очистка кэша таким вот варварским образом – это хорошо.



```
[root@localhost ~]# free
              total        used        free      shared  buff/cache   available
Mem:        2039768      132900     1582496         816      324372     1720744
Swap:       2096124           0     2096124
[root@localhost ~]# df -h
Filesystem      Size  Used Avail Use% Mounted on
devtmpfs        987M     0  987M   0% /dev
tmpfs           996M     0  996M   0% /dev/shm
tmpfs           996M  808K  996M   1% /run
tmpfs           996M     0  996M   0% /sys/fs/cgroup
/dev/sda1        18G  1.3G   17G   7% /
tmpfs           996M  8.0K  996M   1% /tmp
tmpfs           200M     0  200M   0% /run/user/0
tmpfs           200M     0  200M   0% /run/user/1000
[root@localhost ~]# sync; echo 3 > /proc/sys/vm/drop_caches
[root@localhost ~]# free
              total        used        free      shared  buff/cache   available
Mem:        2039768      117852     1808844         816      113072     1771460
Swap:       2096124           0     2096124
[root@localhost ~]# _
```

***Рис. 15.9. До и после ввода команды* `sync; echo 3 > /proc/sys/vm/drop_caches`**

15.4. Команда *fuser*

Команда fuser позволяет узнать, какой процесс открыл тот или иной ресурс, например, файл или сетевой порт. Примеры использования программы:

```
fuser -va 23/tcp
fuser -va /chroot/etc/resolv.conf
```

В первом случае мы получим идентификатор процесса, открывшего TCP-порт 23, во втором – идентификатор процесса, открывшего файл /chroot/etc/resolv.conf. Что делать далее – решать вам, например, можно "убить" этот процесс командой `kill`.

15.5. Планировщики заданий

15.5.1. Планировщик *cron*

Планировщик заданий нужен для периодического выполнения каких-либо заданий. Задания могут быть самыми разнообразными – очистка временного каталога для экономии места, очистка кэша, обновление баз антивируса, запущенного на почтовом сервере и т.д. Никаких ограничений нет – вы можете написать на *bash* небольшой сценарий с необходимыми вам действиями, а затем настроить планировщик для его периодического выполнения.

Самый древний планировщик заданий – *cron*. Он появился еще во времена первых версий UNIX. Но примечательно то, что в современных дистрибутивах используются его модифицированная версия, которая настраивается практически так же. Так, в современных версиях дистрибутивов openSUSE и Fedora используется демон *cronie*, который является потомком того самого демона *cron*.

Не будем вникать в тонкости новой версии, а просто разберемся, как ее настроить. Как и в том самом *cron* есть файл /etc/crontab – это таблица расписания планировщика задач. Формат записей в этом файле следующий:

```
минуты (0-59) часы (0-23) день (1-31) месяц (1-12) день_недели
(0-6, 0 — Вс) команда
```

В листинге 15.1 приведен пример этого файла по умолчанию.

Листинг 15.1. Пример файла /etc/crontab

```
SHELL=/bin/bash
PATH=/usr/bin:/usr/sbin:/sbin:/bin:/usr/lib/news/bin
MAILTO=root
```

```
-*.15 * * * * root  test -x /usr/lib/cron/run-crons && /usr/lib/cron/run-
crons >/dev/null 2>&1
```

> **Примечание**. Просмотреть (даже просмотреть, не говоря уже о модификации!) файл /etc/crontab может только пользователь *root*. Поэтому сначала нужно получить права *root*, а затем уже что-либо делать с файлом /etc/crontab.

Переменная SHELL задает путь к оболочке, PATH – путь поиска программ, а MAILTO определяет имя пользователя, которому будет отправлен отчет о выполнении расписания. В таблице расписания всего одна запись – она проверяет наличие сценариев в каталогах cron.hourly, cron.daily, cron.weekly и cron.monthly и их выполнение. Об этом мы поговорим чуть позже.

Представим, что вам нужно выполнять какой-то сценарий каждый день в 8:30. Для этого в /etc/crontab нужно добавить строку:

```
30 8  *  *  *  /usr/bin/script arguments
```

Однако планировщик предлагает более удобный способ изменения таблицы расписания. Представим, что у вас есть команда, которую нужно выполнять периодически. Создайте файл *myscript* со следующим содержимым:

```
#/bin/bash
/путь/ myscript аргументы
```

Сохраните файл и установите право выполнения для этого файла:

```
chmod +x myscript
```

Только что вы создали простейший сценарий, который просто выполняет вашу команду с заданными аргументами. При желании, необходимости и знании *bash* (информацию по этой оболочке вы без особых проблем найдете в Интернете) вы можете усовершенствовать этот сценарий.

После того, как сценарий создан, его нужно поместить в один из каталогов:

- /etc/cron.hourly – содержит сценарии, которые будут выполняться каждый час;
- /etc/cron.daily – содержит сценарии, который будут выполняться ежедневно;
- /etc/cron.weekly – сюда нужно поместить сценарии, которые будут выполняться еженедельно;
- /etc/cron.monthly – содержит сценарии, которые будут выполнены раз в месяц.

Просто поместите сценарий в один из этих каталогов и положитесь на *cron* – далее вашего вмешательства не требуется.

Также существует возможность создать отдельное расписание для каждого пользователя. Для этого используется команда `crontab`. Однако такая возможность на современных серверах (когда пользователи не работают с терминалом сервера непосредственно) используется довольно редко. При этом в файл /etc/cron.deny заносятся пользователи, которым запрещено использовать планировщик *cron*. Если вам нужно отредактировать таблицу расписания для каждого пользователя, используйте команду `crontab`. В большинстве случаев команда будет такой:

```
crontab -e -u <имя_пользователя>
```

После этого откроется текстовый редактор (определенный в переменной окружения EDITOR) для редактирования таблицы расписания указанного пользователя. Синтаксис такой же, как для общесистемной таблицы расписания.

15.5.2. Планировщик *anacron*

Планировщик *anacron* – еще один форк старого-доброго планировщика *cron*. Ситуация с *anacron* такая: когда стало понятно, что *cron* устарел, начали появляться альтернативные планировщики, подобные ему. У каждого из них были свои преимущества и недостатки, но некоторые разработчики дистрибутивов остановили свой выбор на планировщике *anacron*. Однако прошло время и сейчас этот планировщик практически не используется. В основном используется планировщик *cronie*, который настраивается практически так же, как и *cron*, что и было показано ранее.

Конечно, вам могут встретиться дистрибутивы, в которых по каким-то причинам используется *anacron*. Как правило, это устаревшие версии дистрибутивов. Если же вы обнаружили *anacron* в современной версии дистрибутива, то, скорее всего, это личное предпочтение разработчиков дистрибутивов.

Основное "визуальное" отличие этого планировщика – наличие файла /etc/anacrontab вместо просто /etc/crontab. Формат записей также другой:

```
Период    Задержка         ID             Команда
```

Пример таблицы расписания *anacrontab*:

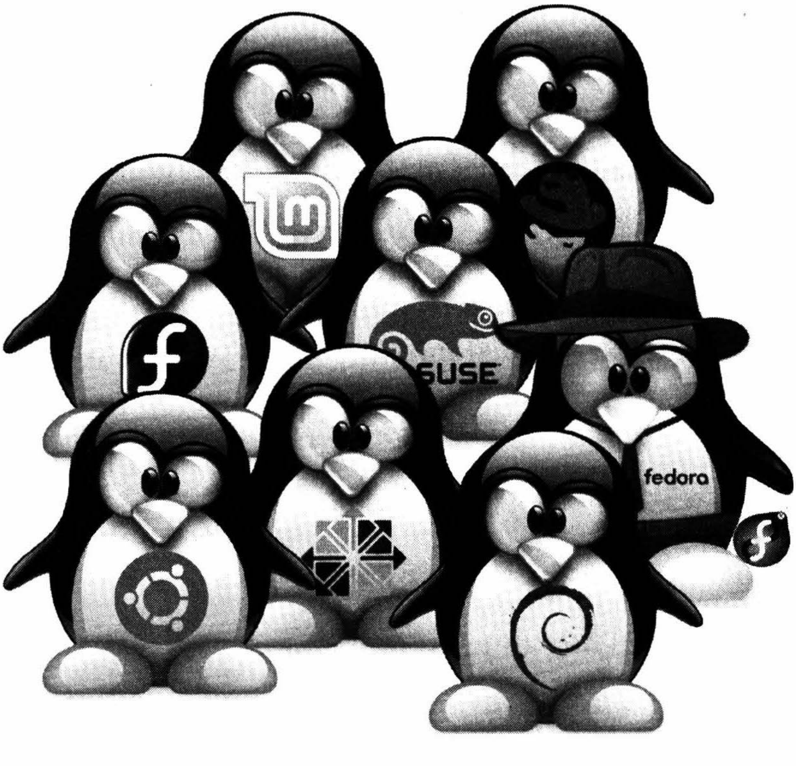
```
1  5    cron.daily        run-parts /etc/cron.daily
7  10   cron.weekly       run-parts /etc/cron.weekly
30 75   cron.monthly      run-parts /etc/cron.monthly
```

Принцип прост: как и в предыдущем случае, вам нужно поместить ваш сценарий в один из каталогов /etc/cron.daily, /etc/cron.weekly или /etc/cron.monthly (каталога cron.hourly для этого планировщика не было предусмотрено).

По сути, как только вы увидели каталоги cron.daily, cron.weekly или cron.monthly, можете сразу помещать в них свои сценарии, особо не разбираясь, какой планировщик установлен. Самое главное, чтобы служба *cron* была включена.

Глава 16.

Маршрутизация и настройка брандмауэра



16.1. Просмотр таблицы маршрутизации

Отправляемые данные, как вы знаете, не отправляются целиком – они разбиваются на меньшие части – пакеты. Маршрутизация – это процесс перенаправления пакета по различным сетям вплоть до места назначения.

В сетях TCP/IP информация маршрутизации хранится в виде таблицы маршрутизации. Таблица маршрутизации содержит ряд простых правил. Например, если пакет отправляется в *сеть 1* он должен быть отправлен на маршрутизатор M1, если пакет отправляется в *сеть 2,* то его нужно отправить на маршрутизатор M2. Пакет, отправляемый на любую другую сеть (не 1 и не 1), нужно отправить на маршрутизатор M (шлюз по умолчанию). Получив пакеты, маршрутизаторы M1, M1 и M сами знают, что с ними делать. Возможно, передать дальше другому маршрутизатору, а может отправить узлу-получателя пакета. Все зависит от такой же таблицы маршрутизации, но уже на тех маршрутизаторах. Что будут делать маршрутизаторы с полученными пакетами – это их дело, наше дело – доставить пакеты до этих маршрутизаторов.

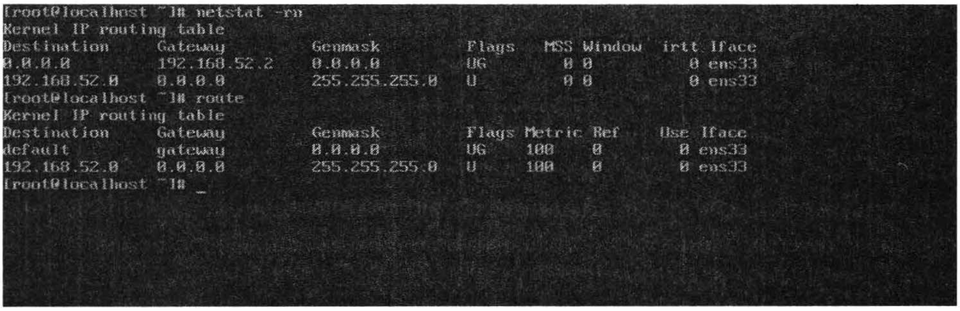
Таблица маршрутизации ядра Linux хранит данные о маршрутизации. Каждая строка в этой таблице содержит несколько параметров – адрес сети назначения, маска сети, флаги, интерфейс и т.д.

Ядро при отправке пакета исследует таблицу маршрутизации: оно определяет, в какую сеть направляется пакет – если она есть в таблице маршрутизации, то пакет отправляется через заданный в таблице интерфейс. Если нужной сети в таблице нет, пакет отправляется или на шлюз по умолчанию или же (если он не задан), отправителю пакета передается ICMP-сообщение «Network Unreachable» (сеть недоступна).

Просмотреть таблицу маршрутизации ядра можно или командой `netstat -rn` или командой `route`:

```
# netstat -rn
# route
```

Вывод команд немного отличается, как видно из рис. 16.1. Здесь видно, что шлюз по умолчанию – 192.168.52.2. По сути, это самая простая таблица маршрутизации, какая только может быть.



```
[root@localhost ~]# netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
0.0.0.0         192.168.52.2    0.0.0.0         UG        0 0          0 ens33
192.168.52.0    0.0.0.0         255.255.255.0   U         0 0          0 ens33
[root@localhost ~]# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         gateway         0.0.0.0         UG    100    0        0 ens33
192.168.52.0    0.0.0.0         255.255.255.0   U     100    0        0 ens33
[root@localhost ~]# _
```

***Рис. 16.1. Команды* `netstat -rn` *и* `route`**

Разберемся, что содержится в столбцах таблицы маршрутизации. Столбец Destination хранит адрес сети назначения, Gateway – шлюз (маршрутизатор), которому нужно отправить пакеты, чтобы они достигли сеть из колонки Destination.

Столбец Genmask содержит маску сети, а Flags – флаги маршрута. Флаги могут быть следующими:

- U — маршрут активен;
- H — маршрут для хоста (H- host), а не для сети;
- G — флаг шлюза (G- gateway);
- D — динамический маршрут, который был установлен демоном маршрутизации;

- M — маршрут, который был модифицирован демоном маршрутизации;
- C — запись кэширована;
- ! — запрещенный маршрут.

Колонка MSS (Maximum Segment Size) содержит значение MSS – максимальный размер сегмента для TCP-соединений по этому маршруту. Столбец **Window** показывает размер окна по умолчанию для TCP-соединений по этому маршруту. Столбец **irtt** — это начальное время RTT. В большинстве сетей время RTT не нужно изменять, но в некоторых медленных сетях время RTT можно увеличить, чтобы избежать лишних повторений пакетов. Система отправляет пакет и ждет некоторое время (RTT) от получателя подтверждения получения. Если подтверждение получения не было, тогда система отправляет пакет еще раз. В медленных сетях подтверждение получения может не успеть дойти до отправителя пакета, поэтому RTT увеличивают (это можно сделать с помощью команды *route*). Столбец **Iface** задает интерфейс, используемый для отправки пакета.

В выводе команды *route* вместо столбцов MSS и Window есть колонки **Metric** и **Ref**. Первая содержит метрику, то есть расстояние до маршрутизатора в *хопах* (переходах): один хоп – это один маршрутизатор. Столбец **Ref** содержит – это количество ссылок на маршрут. Ядром Linux этот параметр не учитывается.

16.2. Изменение и сохранение таблицы маршрутизации

Для редактирования таблицы маршрутизации используется команда *route*. Записи в таблице маршрутизации бывают статическими (добавляются командой *route*) или динамическими (добавляются по мере работы системы демоном маршрутизации).

Вот как можно добавить маршрут по умолчанию командой *route*:

```
# route add default gw 192.168.1.1 eth0
```

Думаю, эта команда понятна: 192.168.1.1 – это новый шлюз по умолчанию, а пакеты к нему будут отправляться через интерфейс eth0.

При перезагрузке таблица маршрутизации очищается, поэтому, если она формируется вручную, то есть командами *route*, а не каким-либо демоном

маршрутизации, то эти команды нужно добавить в конфигурационные файлы сети или сценарии инициализации системы, чтобы внесенные вами изменения не были потеряны.

Шлюз по умолчанию можно сохранить в файле /etc/sysconfig/networking-scripts/ifcfg-имя_интерфейса. В листинге 16.1 приводится пример этого файла.

Листинг 16.1. Файл /etc/sysconfig/networking-scripts/ifcfg-имя_интерфейса

```
DEVICE=eth0
HWADDR=00:0C:29:E8:F0:C4
TYPE=Ethernet
ONBOOT=yes
NETMASK=255.255.255.0
IPADDR=192.168.1.101
GATEWAY=192.168.1.1
NETWORK=192.168.1.0
BROADCAST=192.168.1.255
BOOTPROTO=none
NM_CONTROLLED=no
```

Параметр GATEWAY позволяет указать IP-адрес шлюза. Но обычно вам не придется редактировать файлы настройки сетевых интерфейсов обычных компьютеров, поскольку такая общая информация как IP-адрес шлюза задается DHCP-сервером примерно так:

```
subnet 192.168.1.0 netmask 255.255.255.0 {
    # Список маршрутизаторов (через пробел)
     option routers                  192.168.1.1;
     ...
```

Теперь рассмотрим общий формат вызова команды *route*:

```
# route [операция] [тип] адресат gw шлюз [метрика] [dev интерфейс]
```

Параметр `операция` может принимать значения *add* и *del*. Первый добавляет маршрут, а второй – удаляет. Второй параметр (`тип`) необязательный – он позволяет задать тип маршрута: *default* (маршрут по умолчанию), *-net* (маршрут к сети), *-host* (маршрут к узлу).

Третий параметр, `адресат`, содержит адрес сети, если вы задаете маршрут к сети, или адрес узла, если добавляется маршрут к узлу). Если вы задаете маршрут по умолчанию, то этот параметр вообще не нужно указывать.

Параметр `шлюз` задает IP-адрес шлюза. Можно также указать и доменное имя, но обычно указывается IP-адрес. Параметр `метрика` задает число переходов (маршрутизаторов) на пути к адресату. Параметр `dev` нужно указывать, если в системе установлено несколько сетевых интерфейсов и нужно указать, через какой именно сетевой интерфейс нужно отправить пакеты. Оба последних параметра не являются обязательными.

Рассмотрим несколько примеров использования команды *route*:

```
# route add -net 192.168.16.0 netmask 255.255.255.0 dev ens33
# route add -net 192.168.16.0 netmask 255.255.255.0 gw 192.168.16.1
# route add -net 10.100.0.0 netmask 255.0.0.0 reject
# route del 10.100.0.0
```

Первая команда добавляет маршрут к сети 192.168.16.0 через устройство **ens33**. Как видите, мы не указываем IP-адрес шлюза, а просто указали имя интерфейса – все пакеты, адресованные сети 192.168.16.0, будут отправлены через интерфейс ens33.

Вторая команда добавляет маршрут к сети 192.168.16.0 через шлюз 192.168.16.1. Все пакеты, адресованные этой сети, будут отправлены маршрутизатору с IP-адресом 192.168.16.1. В этом случае сетевой интерфейс указывать не обязательно.

Третья команда задает запрещающий маршрут. Отправка пакетов в сеть 10.100.0.0 запрещена (параметр *reject*). Последняя команда удаляет ранее заданный запрещающий маршрут.

Удаление маршрутов в Linux заслуживает отдельного разговора. Команда удаления маршрута выглядит так:

```
# route del IP-адрес
```

В Linux нет параметра *-f* (как в FreeBSD), который позволяет удалить сразу все маршруты (то есть очистить таблицу маршрутизации), поэтому вам придется ввести ряд команд *route.*

Как уже было отмечено, таблица маршрутизации очищается при перезагрузке. Чтобы этого не произошло, маршруты нужно определить в файлах конфигурации сети.

В Debian/Astra Linux/старых версиях Ubuntu настройка статических маршрутов производится в файле /etc/network/interfaces. Просто добавьте в секцию настройки интерфейса команду *up* и укажите команду *route*, которую нужно выполнить. Допустим, для настройки маршрутов вы ввели три команды:

```
route add -net 192.168.1.0 netmask 255.255.255.0 gw 192.168.17.254 eth0
route add -net 192.168.14.0 netmask 255.255.255.0 gw 192.168.17.254 eth0
route add -net 192.168.22.0 netmask 255.255.255.0 gw 192.168.17.254 eth0
```

Тогда в файл /etc/network/interfaces нужно добавить следующие строки:

```
up route add -net 192.168.1.0 netmask 255.255.255.0 gw 192.168.17.254 eth0
up route add -net 192.168.12.0 netmask 255.255.255.0 gw 192.168.17.254 eth0
up route add -net 192.168.21.0 netmask 255.255.255.0 gw 192.168.17.254 eth0
```

В этом примере пакеты ко всем трем сетям направляются на шлюз 192.168.17.254, а он уже сам решает, что с ними делать. Прежде, чем мы рассмотрим настройку брандмауэра, нужно отметить, как превратить компьютер в шлюз. Для этого нужно разрешить пересылку пакетов протокола IPv4 (IPv4 forwarding). Для этого добавьте значение *1* в файл /proc/sys/net/ipv4/ip_forward:

```
# echo 1 > /proc/sys/net/ipv4/ip_forward
```

Конечно, после перезапуска это значение будет потеряно. Чтобы его сохранить, добавьте в файл /etc/sysctl.conf следующую строку:

```
net.ipv4.ip_forward=0
```

Итак, ваш компьютер теперь превращен в шлюз. Но сам по себе перенаправлять пакеты он не будет. Нужно задать ряд правил перенаправления пакетов. Для этого мы будем использовать брандмауэр **iptables**, который имеется во всех современных дистрибутивах. Именно о нем и пойдет речь в следующем

разделе. Забегая наперед, скажу, что **iptables** можно использовать не только для настройки шлюза, но и просто для защиты сервера или рабочей станции от нежелательных подключений.

В новых версиях Ubuntu (начиная с версии 18.04) настройка маршрутов осуществляется через файл /etc/netplan/01-netcfg.yaml. Откройте этот файл:

```
sudo nano /etc/netplan/01-netcfg.yaml
```

В нем вы найдете конфигурацию по умолчанию, которая может быть примерно такой:

```
# This file describes the network interfaces available on your
system
# For more information, see netplan(5).
network:
  version: 2
  renderer: networkd
  ethernets:
    eno1:
      dhcp4: yes
```

Добавьте следующие строки:

```
      routes:
      - to: 192.168.44.0/24
        via: 192.168.0.1
```

Данная конфигурация означает, что маршрут к сети 192.168.44.0/24 (маска 255.255.255.0) будет проходить через маршрутизатор 192.168.0.1. Полная конфигурация будет выглядеть так:

```
# This file describes the network interfaces available on your
system
# For more information, see netplan(5).
network:
  version: 2
  renderer: networkd
  ethernets:
```

```
    eno1:
      dhcp4: true
      routes:
      - to: 192.168.44.0/24
        via: 192.168.0.1
```

Обратите внимание, что YAML-файл очень требователен к отступам и разметке. Убедитесь, что оператор «routes» находится на расстоянии двух пробелов от имени интерфейса (в нашем случае *eno1*), к которому вы применяете маршрут.

Сохраните изменения в файле и примените их посредством команды:

```
sudo netplan apply
```

Проверим, все ли хорошо:

```
ip route show
```

Убедитесь, что вы видите эту строку в выводе предыдущей команды:

```
192.168.44.0/24 via 192.168.0.1 dev eno1 proto static
```

Если вы не видите данный статический маршрут в выводе, значит, что-то не так с вашей конфигурацией. Вернитесь к YAML-файлу и проверьте разметку/отступы. Вы также можете проверить конфигурацию командой:

```
sudo netplan try
```

16.3. Настройка брандмауэра iptables

Брандмауэр **iptables** на данный момент считается устаревшим, но до сих пор используется даже в самых новых версиях дистрибутивах. От него пытаются отказаться последние лет 5, но до сих пор этого не произошло и, на наш взгляд, в ближайшие лет 5 он будет все еще актуален. В следующем разделе мы рассмотрим современный брандмауэр **ifw**, который, судя по всему, вытеснит **iptables** в будущем, хотя на данный момент он даже не устанавливается по умолчанию.

16.3.1. Преобразование сетевого адреса

Что такое брандмауэр, я надеюсь, объяснять не нужно. А вот о преобразовании *сетевого адреса* (NAT, Network Address Translation) поговорить стоит, чтобы не было лишних вопросов. Пространство IP-адресов не безразмерное. Рано или поздно IPv4-адреса закончатся. Именно поэтому и был разработан протокол IPv6. Учитывая тенденции роста Интернета, думаю, что переход на IPv6 состоится в ближайшие годы, но пока будем говорить исключительно о IPv4, поскольку новый протокол IPv6 пока практически не используется.

Если бы реальные IP-адреса раздавались бы всем желающим, они бы уже давно закончились. Поэтому обычно при подключении к Интернету клиенту выдается всего один IP-адрес, даже если у вас целая организация. Этот один реальный IP-адрес, как правило, используется на шлюзе. А во внутренней сети используются локальные IP-адреса: 10.*.*.* (сеть класса А), 172.16.*.*–172.31.*.* (класс В) и 192.168.*.* (сеть класса С). Эти IP-адреса не могут пройти через маршрутизатор Интернета – как только маршрутизатор увидит локальный IP-адрес, пакет сразу же будет отброшен. Но зато такие IP-адреса вполне подойдут для использования в локальных сетях. Данные IP-адреса уникальны только в пределах вашей организации. В соседней организации могут использоваться такие IP-адреса, как и у вас, но поскольку между компьютерами организаций нет взаимодействия, в этом нет ничего страшного.

Когда компьютер с локальным IP-адресом 192.168.1.102 отправляет пакет компьютеру, обладающему реальным IP-адресом, например, веб-серверу www.example.com, наш шлюз должен выполнить NAT. То есть он должен перезаписать локальный IP-адрес так, чтобы он выглядел как реальный. Поскольку в нашем распоряжении только один реальный IP-адрес (IP-адрес шлюза), то веб-сервер www.example.com будет «думать», что запрос пришел от нашего шлюза, и он ничего не будет подозревать о компьютерах, которые находятся за шлюзом. Веб-сервер обработает запрос и отправит пакет. Наш шлюз получит этот пакет (с ответом) и отправит его компьютеру 192.168.1.102, перезаписав поле отправителя так, что компьютер 192.168.1.102 будет «думать», что ответ пришел непосредственно от www.example.com.

16.3.2. Цепочки и правила

При настройке брандмауэра **iptables** очень важно разобраться с цепочками и правилами. Ведь основная задача брандмауэра – это фильтрация пакетов, проходящих через сетевой интерфейс. Когда брандмауэр получает пакет, он анализирует его и затем принимает решение – принять его или уничтожить.

Брандмауэр может выполнять и более сложные действия, но обычно достаточно этих двух.

Обработка пакета заключается в его прохождении по цепочке правил. Каждое правило состоит из условия и действия. Если пакет соответствует условие, то выполняется указанное действие. Действие также называется целью. Если пакет не соответствует условию правила, то он передается следующему правилу. Если же пакет не соответствует ни одному из правил цепочки, выполняется действие по умолчанию.

Цепочки правил существует не сами по себе, а собираются в таблицы. Таблица *filter* является основной таблицей, отвечает за фильтрацию пакетов, в ней содержатся правила фильтрации пакетов. Таблица *nat* используется при преобразовании сетевого адреса. Есть еще одна таблица – *mangle*, которая используется, если нужно произвести специальные действия над пакетом, отличные от фильтрации и NAT.

В состав каждой таблицы входят три цепочки: INPUT, OUTPUT и FORWARD. Первая используется для входящих пакетов, вторая – для исходящих, а третья для пересылаемых пакетов. При желании вы можете создать собственную таблицу, но в ней все равно будут цепочки INPUT, OUTPUT и FORWARD.

Теперь поговорим о действиях над пакетом. Действие ACCEPT принимает пакет, DROP – уничтожает пакет. Действие MASQUERADE позволяет скрыть IP-адрес пакета. Если же в качестве действия указано имя цепочки, то пакет будет отправлен для обработки в указанную цепочку.

Рассмотрим процесс обработки входящего пакета. Сначала пакет поступает в цепочку PREROUTING таблицы *mangle*. После чего, если он не был уничтожен правилами таблицы *mangle*, он попадает в цепочку PREROUTING таблицы *nat*. Здесь правила проверяют, нужно ли модифицировать назначение пакета или нет. После этого пакет направляется или в цепочку FORWARD (если его нужно передать дальше – другому компьютеру) или в цепочку INPUT (если пакет адресован этому компьютеру).

Если пакет был адресован этому компьютеру, то он передается правилам цепочки INPUT таблицы *mangle* и *filter*. Если эти правила не уничтожили пакет, то он отправляется приложение, например, веб-серверу. Приложение получает данные, обрабатывает их и отправляет ответ. Ответ приложения преобразуется в пакет, который сначала обрабатывается цепочкой OUTPUT таблиц *mangle*, *nat* и *filter*, а затем отправляется в цепочку POSTROUTING и обрабатывается правилами таблиц *mangle* и *nat*. Если после всего этого пакет еще жив, он становится исходящим пакетом и отправляется в сеть.

16.3.3. Команда *iptables*

Для управления правилами брандмауэра используется команда *iptables*. У этой команды очень и очень много различных параметров. В этой книге не будет полного руководства по *iptables*, потому что есть документация (*man iptables*), доступная каждому пользователю. Вместо переписывания руководства другими словами мы остановимся на практическом применении *iptables*. Сначала рассмотрим часто используемые параметры (чтобы вы понимали, что происходит), а затем настроим шлюз для небольшой организации.

Чтобы добавить правило в цепочку, нужно использовать параметр *-A*:

```
# iptables -A <цепочка> <правило>
```

По умолчанию правило будет добавлено в таблицу *filter*. Если нужно указать другую таблицу, то для этого используется параметр *-t*:

```
# iptables -t <таблица> -A <цепочка> <правило>
```

Параметр -P позволяет задать действие по умолчанию:

```
# iptables -P INPUT DROP
# iptables -P FORWARD ACCEPT
# iptables -P OUTPUT DROP
```

В таблице 16.1 указываются возможные действия, которые вы можете использовать с **iptables**.

Таблица 16.1. Возможные действия над пакетами

Действие	Описание
ACCEPT	Принимает пакет и передает его дальше – в следующую цепочку
DROP	Уничтожает пакет, как будто бы его никогда не было
REJECT	Уничтожает пакет, а отправителю пакета сообщается об этом с помощью ICMP-сообщения. Параметр *--reject-with* позволяет уточнить тип ICMP-сообщения: icmp-host-unreachable — узел недоступен; icmp-net-unreachable — сеть недоступна; icmp-port-unreachable — порт недоступен; icmp-proto-unreachable — протокол недоступен

LOG	Протоколирует информацию о пакете в протокол. Полезно из соображений отладки, когда вы настраиваете шлюз
RETURN	Возвращает пакет в цепочку, откуда он прибыл. Использовать не рекомендуется, поскольку возможны зацикливания – вы можете легко попасть в бесконечный цикл, если будете использовать это действие
SNAT	Выполняет подмену IP-адреса источника (Source NAT, SNAT). Данное действие используется в цепочках POSTROUTING и OUTPUT таблицы *nat*
DNAT	Выполняет подмену IP-адреса получателя (Destination NAT, DNAT). Поддерживается только в цепочке POSTROUTING таблицы *nat*
MASQUERADE	Используется в таблице POSTROUTING таблицы *nat*. Чем-то похоже на SNAT, но используется при работе с динамическими IP-адресами, когда возможна «потеря» интерфейса при изменении IP-адреса

Прежде, чем мы начнем создавать наш собственный шлюз, нужно рассмотреть параметры, касающиеся фильтрации пакетов (табл. 16.2).

Таблица 16.2. Параметры, касающиеся фильтрации пакетов

Параметр	Описание
--source	Указывает источник пакета. Вы можете указывать, как доменное имя, так и IP-адрес или даже целую подсеть, например, 192.168.5.0/255.255.255.0
--destination	Указывает получателя пакета. Аналогично, можно указывать, как доменное имя, так и IP-адрес
--protocol (-p)	Позволяет указать протокол. Вы можете использовать идентификаторы, определенные в файле /etc/protocols
--source-port (--sport)	Указывает порт отправителя. Опцию можно использовать вместе с параметром *-p*
--destination-port (--dport)	Указывает порт получателя. Можно использовать вместе с параметром *-p*

--state	Позволяет указать состояние пакета. Фильтр по состоянию возможен только при загрузке модуля state (-m state). Возможны следующие состояния пакета: NEW (новое соединение), ESTABLISHED (установленное соединение), RELATED (связанные с соединением пакеты), INVALID (неопознанные пакеты)
--in-interface (-i)	Указывает входящий интерфейс
--out-interface (-o)	Указывает исходящий интерфейс

В таблице 16.2 указаны далеко не все параметры, но для организации собственного шлюза представленных параметров будет более чем достаточно.

16.3.4. Практический пример

Сейчас рассмотрим практический пример – настройку шлюза небольшой сети с нуля на базе дистрибутива Debian. У нас есть следующие вводные данные:

- Внешняя сеть – *интерфейс eth0*, реальный IP-адрес 88.99.88.99 (IP-адрес приведен для примера).
- Внутренняя – *интерфейс eth1*, IP-адрес сети 192.168.1.0/24 (у вас, скорее всего, будет другой адрес).

Простым «перебросом» пакетов мы не ограничимся, а постараемся решить попутные задачи, а именно:

- Запретить доступ сотрудников к некоторым сайтам. Администрации доступ к этим сайтам будет разрешен.
- Запретить ICQ для сотрудников, но не для администрации
- Запретить некоторые нежелательные приложения.
- Открыть доступ к порту 80 извне для доступа к будущему веб-сайту. Все, что вам останется сделать – это установить nginx (или Apache) и ваш сайт будет виден из Интернета.
- Открыть доступ к шлюзу извне для его администрирования по SSH.
- Открыть порты 25 и 110, необходимые для работы будущего почтового сервера.

- Администратору (его IP-адрес 192.168.1.10) открываем все, что ему будет нужно.

Создайте каталог /etc/firewall, в котором мы будем хранить все необходимое для организации нашего шлюза. Сейчас создайте в ней два файла – admins.txt и black.txt. В первый нужно внести IP-адреса администрации (по одному IP-адресу в одной строке) – им будут доступны все сайты. Это может быть сотрудники IT-отдела, директор, его зам и т.д. – в общем, все вышестоящее начальство. В файл black.txt нужно внести сайты, доступ к которым нужно ограничить (тоже по одному сайту в одной строке).

```
# mkdir /etc/firewall
# touch /etc/firewall/admins.txt
# touch /etc/firewall/black.txt
```

После этого можно приступить, собственно, к настройке шлюза. Первым делом нам нужно отключить Network Manager и настроить наши интерфейсы статически. На сервере (шлюзе) Network Manager не нужен, а сетевые интерфейсы можно легко настроить с помощью /etc/network/interfaces.

Сначала введите команду:

```
# runlevel
```

Вы узнаете текущий уровень запуска:

```
# runlevel
N 2
```

Далее переходим в каталог /etc/rcX.d, где X – это уровень запуска и удаляем ссылку на *network-manager*. Можно также использовать команду:

```
# update-rc.d -f network-manager remove
```

Далее остановите Network Manager:

```
# /etc/init.d/network-manager stop
```

После этого отредактируйте файл /etc/network/interfaces и сконфигурируйте сетевые интерфейсы (лист. 16.2).

Листинг 16.2. Файл /etc/network/interfaces

```
# Локальный интерфейс lo
auto lo
iface lo inet loopback

# Внешняя сеть
auto eth0
iface eth0 inet static
address 88.99.88.99
netmask 255.255.255.0
network 88.99.88.99
dns-nameservers 8.8.8.8

# Внутренняя сеть
auto eth1
iface eth1 inet static
address 192.168.1.1
netmask 255.255.255.0
network 192.168.1.0
broadcast 192.168.1.255
```

Перезапустим сеть

```
# service networking restart
```

Теперь создайте каталог /etc/firewall и поместите туда файл firewall.sh, который будет содержать команды, превращающий наш обычный компьютер в шлюз:

```
# touch /etc/firewall/firewall.sh
# chmod +x /etc/firewall/firewall.sh
# nano /etc/firewall/firewall.sh
```

Код файла firewall.sh приведен в листинге 16.3.

Листинг 16.3. Файл firewall.sh

```
#!/bin/bash

# Определяем некоторые переменные, чтобы облегчить редактирование
# конфигурации в будущем
# Внешний IP-адрес и внешний интерфейс
WAN_IP1=88.99.88.99
WAN_IFACE1=eth0
```

```
# Внутренний адрес сервера и внутренний интерфейс
LAN_IP=192.168.1.1
LAN_IFACE=eth1

# Внутренняя сеть
LOCAL_NET=192.168.1.0/24

# loopback
LO_IFACE=lo
LO_IP=127.0.0.1

# Путь к iptables
ip=/sbin/iptables

# Черный список
blacklist=( $(cat "/etc/firewall/black.txt") )
admins=( $(cat "/etc/firewall/admins.txt") )

# Очищаем все таблицы iptables
$ip -F -t filter
$ip -F -t nat
$ip -F -t mangle
$ip -F

# Правила по умолчанию: запрещаем все, что явно не разрешено
$ip -P INPUT DROP
$ip -P OUTPUT DROP
$ip -P FORWARD DROP

# Превращаем компьютер в шлюз, на всякий случай, если мы забыли сделать
# это раньше
echo 1 > /proc/sys/net/ipv4/ip_forward

# Включаем NAT, чтобы локальные узлы могли получать доступ к Интернету
$ip -t nat -A POSTROUTING -o $WAN_IFACE1 -s $LOCAL_NET ! -d
$LOCAL_NET -j SNAT --to-source $WAN_IP1

# Отбрасываем INVALID-пакеты
$ip -A INPUT   -m state --state INVALID -j DROP
$ip -A FORWARD -m state --state INVALID -j DROP

# Разрешаем трафик через loopback
$ip -A INPUT -p all -i $LO_IFACE -j ACCEPT
$ip -A OUTPUT -p all -o $LO_IFACE -j ACCEPT

# Разрешаем трафик через внутренний адаптер
# Между сервером (шлюзом) и локальной сетью разрешаем все
```

```
$ip -A INPUT -p all -i $LAN_IFACE -s $LOCAL_NET --match state --state
NEW,ESTABLISHED -j ACCEPT

# Разрешаем исходящие новые и уже установленные соединения
# в внутреннюю сеть с адаптера локальной сети
$ip -A OUTPUT -p all -o $LAN_IFACE -d $LOCAL_NET --match state --state
NEW,ESTABLISHED -j ACCEPT

# Разрешаем новые и уже установленные соединения извне (с внешней сети)
# к портам 80 (веб-сервер) и 22 (ssh):

$ip -A INPUT -p tcp -i $WAN_IFACE1 -m multiport --dports
80,22,25,110 --match state --state NEW,ESTABLISHED -j ACCEPT
$ip -A OUTPUT -p tcp -o $WAN_IFACE1 -m multiport --sports
80,22,25,110 --match state --state ESTABLISHED,RELATED -j ACCEPT

# Разрешаем выход с сервера во внешнюю сеть, но только на определенные порты
# Разрешаем порты 80 (HTTP), 443 (SSL) и 53 (DNS)
$ip -A INPUT -i $WAN_IFACE1 -p tcp -m multiport --sports 80,53,443 -j ACCEPT
$ip -A OUTPUT -o $WAN_IFACE1 -p tcp -m multiport --dports 80,53,443 -j ACCEPT
$ip -A INPUT -i $WAN_IFACE1 -p udp -m multiport --sports 53 -j ACCEPT
$ip -A OUTPUT -o $WAN_IFACE1 -p udp -m multiport --dports 53 -j ACCEPT

# Открываем админу (192.168.1.10) доступ ко всему, что будет нужно
# Просмотрите список портов и откройте то, что вам будет нужно
# tcp

$ip -A FORWARD -p tcp -s 192.168.1.10 ! -d $LOCAL_NET -m multiport
--dports 80,53,443,22,25,110,5190 -j ACCEPT
$ip -A FORWARD -p tcp -d 192.168.1.10 ! -s $LOCAL_NET -m multiport
--sports 80,53,443,22,25,110,5190 -j ACCEPT

# udp
$ip -A FORWARD -p udp -s 192.168.1.10 ! -d $LOCAL_NET -m multiport --dports
53 -j ACCEPT
$ip -A FORWARD -p udp -d 192.168.1.10 ! -s $LOCAL_NET -m multiport --sports
53 -j ACCEPT

# Резрешаем ICQ только администраторам
i=0
for i in "${admins[@]}"
do
  $ip -A FORWARD -p tcp -d $i --sport 5190 -j ACCEPT
  $ip -A FORWARD -p tcp -s $i --dport 5190 -j ACCEPT
done

# Резрешаем избранным (список admins) доступ к сайтам из черного списка
j=0
for j in "${blacklist[@]}"
```

```
do
 i=0
 for i in "${admins[@]}"
 do
 $ip -A FORWARD -d $i -s $j -j ACCEPT
 done
done

# Всем остальным запрещаем доступ к сайтам из списка blacklist
i=0
for i in "${blacklist[@]}"
do
$ip -A FORWARD -s $i -j DROP
done


# Разрешаем транзит некоторых пакетов (80, 443 и 53)
$ip -A FORWARD -p tcp -s $LOCAL_NET ! -d $LOCAL_NET -m multiport --dports 
80,53,443 -j ACCEPT
$ip -A FORWARD -p tcp -d $LOCAL_NET ! -s $LOCAL_NET -m multiport --sports 
80,53,443 -j ACCEPT
$ip -A FORWARD -p udp -s $LOCAL_NET ! -d $LOCAL_NET -m multiport --dports 
53 -j ACCEPT
$ip -A FORWARD -p udp -d $LOCAL_NET ! -s $LOCAL_NET -m multiport --sports 
53 -j ACCEPT
```

После этого нужно обеспечить автоматический запуск нашего сценария /etc/firewall/firewall.sh.

16.4. Настройка брандмауэра *ufw*

Традиционно в качестве брандмауэра (фильтра пакетов) в Ubuntu используется **iptables**, но поскольку Ubuntu позиционируется как простой дистрибутив, то и оболочка для **iptables** была разработана соответствующая – UTF (Uncomplicated Firewall) – несложный файрвол.

16.4.1. Проверяем состояние брандмауэра

Первым делом нужно убедиться, что пакет **ufw** вообще установлен или установить его, если это не так:

```
sudo apt install ufw
```

Как показано на следующей иллюстрации, уже установлена последняя версия пакета **ufw**.

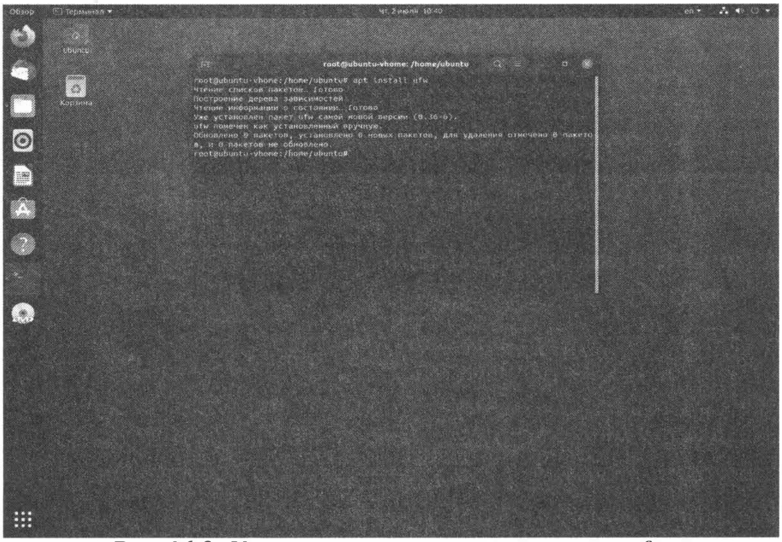


Рис. 16.2. Установлена самая новая версия ufw

Теперь посмотрим состояние брандмауэра:

```
# ufw status verbose
```

По умолчанию фильтр пакетов выключен, поэтому вы получите сообщение

```
Состояние: неактивен
```

Не нужно спешить включать файрвол: сначала его нужно настроить. Ведь если порт 22 окажется по умолчанию недоступен, то вы потеряете доступ к своему серверу, если администрируете его удаленно. Об этом нужно помнить!

16.4.2. Базовая настройка

По умолчанию брандмауэр запрещает все входящие соединения и разрешает все исходящие. Такая политика идеальная с точки зрения безопасности (дальше вы поймете почему) – ведь если кто-то (и вы в том числе) захочет к нему подключиться, что у него это не получится. В то же время приложения на сервере смогут создавать исходящие соединения.

Рассмотрим две команды:

```
ufw default deny incoming
ufw default allow outgoing
```

Данные два правила как раз и задают политику по умолчанию – запрещаются все входящие соединения и разрешаются все исходящие.

Итак, все входящие соединения запрещены. Чтобы к серверу можно было «достучаться» по определенному порту, его нужно сначала открыть. UFW хорош тем, что вам даже не нужно помнить номер порта – нужно знать только название сервиса. Например, вот как можно разрешить подключение по SSH:

```
ufw allow ssh
```

При этом UFW сам создаст правило для порта 22 – именно этот порт используется для SSH. Брандмауэр знает порты и имена всех распространенных служб (http, ssh, ftp, sftp и т.д.).

Однако если вы перенастроили **ssh** на нестандартный порт из соображений той же безопасности, нужно явно указать номер порта:

```
ufw allow 3333
```

После разрешения **ssh** (это главное, чтобы сейчас файрвол нам не разорвал соединение) можно включить **ufw** командой:

```
ufw enable
```

Вы увидите сообщение о том, что межсетевой экран включен и будет запускаться при загрузке системы.

Посмотрите на следующий скриншот (рис. 16.3).

Посмотрим, что произошло. Сначала мы разрешили **ssh**, на что получили ответ, что правила обновлены:

```
Правила обновлены
Правила обновлены (v6)
```

Затем мы включаем файрвол и получаем сообщение, что он активен и будет запускаться при загрузке системы.

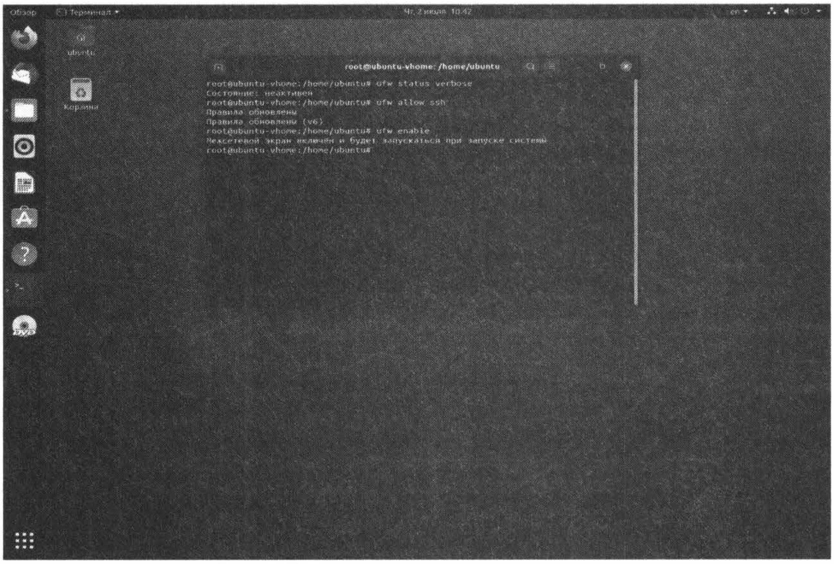


Рис. 16.3. Процесс настройки ufw

На этом базовая настройка выполнена – **ssh** успешно работает, и мы можем приступить к дальнейшей настройке фильтра пакетов.

16.4.3. Создаем правила для других приложений

Теперь нужно разрешить работу других приложений. Как правило, нужно разрешить службу *http* (веб-сервер), *ftp* и постараться не забыть о *https* (что очень важно в последнее время):

```
ufw allow http
ufw allow https
ufw allow ftp
```

Сделать то же самое можно было бы и по номерам портов:

```
ufw allow 80
ufw allow 443
ufw allow 21
```

При желании можно разрешить целый диапазон портов, указав при этом транспортный протокол (UDP или TCP):

```
sudo ufw allow 2000:2200/tcp
sudo ufw allow 4000:4400/udp
```

16.4.4. Разрешаем IP-адреса

Ufw позволяет разрешить определенному IP-адресу доступ ко всем портам сервера, например:

```
ufw allow from 46.229.220.16
```

Если нужно разрешить доступ конкретному IP-адресу только к определенному порту, то делается это так:

```
ufw allow from 46.229.220.16 to any port 22
```

Здесь мы разрешаем не все подключения к **ssh**, а только подключения с IP-адреса 46.229.220.16.

Разрешить доступ целого диапазона IP-адресов (например, когда у администратора динамический IP), можно так:

```
ufw allow from 123.45.67.89/24 to any port 22
```

16.4.5. Запрещаем IP-адреса и службы

Запретить доступ с определенного IP-адреса можно аналогично:

```
ufw deny from 123.45.67.89
```

При желании можно запретить все подключения к определенной службе:

```
ufw deny ftp
```

16.4.6. Удаление/сброс правил

Сбросить все правила можно командой:

```
ufw reset
```

Но убедитесь, что на момент ввода этой команды вы отключили файрвол, иначе вы потеряете доступ по **ssh**.

Удалить конкретное правило можно по номеру. Сначала введите следующую команду, чтобы узнать номер правила:

```
ufw status numbered
```

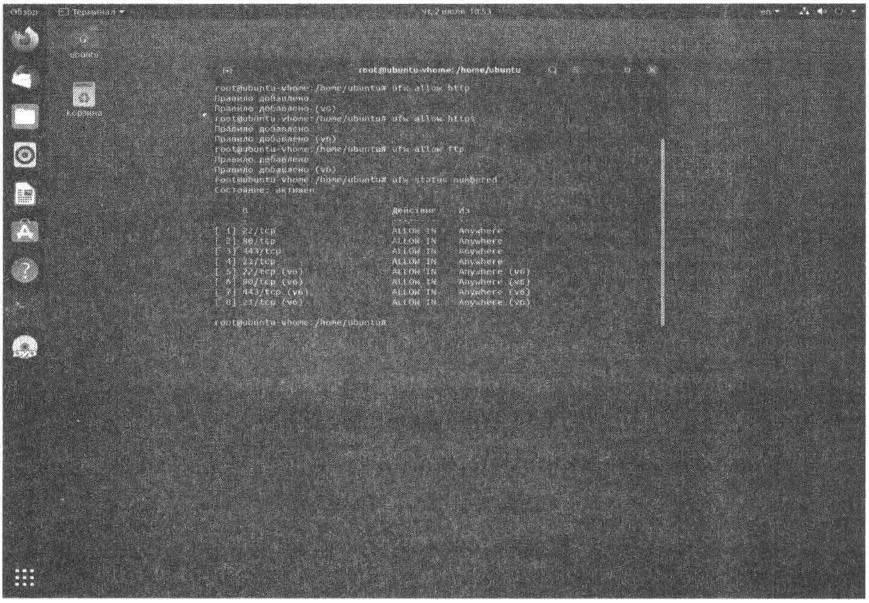


Рис. 16.4. Список правил

16.4.7. Отключение файрвола

Отключить **ufw** можно командой *ufw disable*. Отключение может понадобиться, если какие-то из правил работают неправильно и нужно временно отключить файрвол, чтобы разрешить работу тех или иных сервисов

Если вы ранее использовали **iptables**, то наверняка заметили, что синтаксис **ufw** гораздо проще. Если же до этого вам не приходилось настраивать брандмауэр, то **ufw** – оптимальное решение, с которым не составит труда разобраться даже начинающему администратору.

Колисниченко Д. Н.

LINUX
НА ПРИМЕРАХ

Практика, практика и только практика

Группа подготовки издания:

Зав. редакцией компьютерной литературы: *М. В. Финков*

Редактор: *Е. В. Финков*

Корректор: *А. В. Громова*

12+

ООО "Издательство Наука и Техника"

ОГРН 1217800116247, ИНН 7811763020, КПП 781101001

192029, г. Санкт-Петербург, пр. Обуховской обороны, д. 107, лит. Б, пом. 1-Н

Подписано в печать 18.02.2022. Формат 70x100 1/16.

Бумага газетная. Печать офсетная. Объем 20 п.л.

Тираж 1750. Заказ № 1034/22.

Отпечатано в АО «Можайский полиграфический комбинат»

143200, Россия, г. Можайск, ул. Мира, 93.

www.oaompk.ru, тел.: (49638) 20-685

Колисниченко Д. Н.

Linux НА ПРИМЕРАХ

ПРАКТИКА, ПРАКТИКА И ТОЛЬКО ПРАКТИКА

Данная книга является практическим руководством по работе в Linux и ее администрированию. Книга содержит в себе как теоретические, так и практические материалы, т.е. теория и практика объединены в одно целое. Изложение ведется с учетом самых разных дистрибутивов Linux.

Будет рассмотрен широкий спектр задач и возможностей Linux – от самых основ (установка системы, вход и завершение работы, настройка системы, основы командной строки) до более продвинутых тем (локальное администрирование в Linux; управление файловой системой; маршрутизация и настройка брандмауэра; системные процессы и т.д.).

Книга будет полезна как для тех, кто только заинтересовался Линуксом, так и для тех, кто хочет расширить свои навыки использования этой операционной системой.

ISBN 978-5-94387-410-9

9 78-5-94387-410-9

"Издательство Наука и Техника"
г. Санкт-Петербург

Для заказа книг:
(812) 412-70-26
e-mail: nitmail@nit.com.ru
www.nit.com.ru